DLR-IB-AS-BS-2021-136

Development of a module for mission analysis for a gradient-based aerodynamic shape optimization process

Forschungsbericht

Autor
Javed Butt

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Bericht des Instituts für Aerodynamik und Strömungstechnik
Report of the Institute of Aerodynamics and Flow Technology

DLR-IB-AS-BS-2021-136

Development of a module for mission analysis for a gradient-based aerodynamic shape optimization process

Javed Butt

Herausgeber:
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Aerodynamik und Strömungstechnik
Lilienthalplatz 7, 38108 Braunschweig

ISSN 1614-7790

Stufe der Zugänglichkeit: 1
Braunschweig, im September 2021

Institutsdirektor:
Prof. Dr.-Ing. habil. C.-C. Rossow

Verfasser:
Javed Butt

Abteilung: Center of Computer Applications in Aerospace Science and Engineering

Abteilungsleiter:
Prof. Dr. S. Görtz

Der Bericht enthält:
70 Seiten
84 Bilder
7 Tabellen
31 Literaturstellen

Acknowledgments

All praise and thanks to the **ONE** to Whom all praise, thanks and gratitude belongs.

My special thanks goes to Andrei Merle, who was the supervisor of this work. He showed his knowledge in a modest way, supported me actively in getting this work completed by helping me solving equations, reviewing my work and providing beneficial tricks and tips. Especially, when it came to solving equations, I really appreciated his help. Thank you for your time and effort you spend on this work Andrei Merle.

Also, Professor Stefan Görtz - At each end of our conversations, he would ask me to call him, if any problems or questions would arise - thank you for your support Professor Görtz.

Declaration of independent authorship

I hereby declare that the present work, the Studienarbeit, is solely and independently done by myself in all aspects, such as developments, code implementations, and writing of report. In addition, I confirm that I did not use any tools, materials or sources other than those explicitly specified.

Full name: Javed Butt

Date and place: 19.08.2021, Braunschweig

Signature:

Contents

1 Introduction

This work is written for the *Technische Universiät Braunschweig* in a cooperation with the German Aerospace Center (*DLR*). It is the result of research topic defined by my supervisor Andrei Merle and Prof. Stefan Görtz, both working for the DLR. The overall goal, in very brief terms, is to calculate the fuel consumption and its gradients with respect to later introduced shape parameters for one or multiple individual flight missions. The developed tool for that purpose is called *missioninformer*. The workflow of the *missioninformer* can be explained with figure 1.1. *Missioninformer* can be treated as a black box, which makes its implementation into existing analysis and optimization processes easier. It receives an input, the so-called *mission input*, which contains descriptive information about a flight mission, e.g. payload, cruise range, flight Mach number (a more detailed explanation will be given in section 1.1) and aerodynamical data. With these inputs *missioninformer* provides the user with 2 outputs. The first one is a state value, it is the amount of fuel in kg which is required for the mission. The second output are the gradients of the fuel for the mission with respect to arbitrary aircraft shape parameters.

1.1 Motivation

Aviation has become an indispensable part of global economy. Its importance is evident in the private sector, where it is connecting destinations worldwide and in the business sector, where goods are transported and business trips are undertaken. Since an aircraft burns fuel and thus emits carbon dioxide (CO2), nitrogen oxide (NOx) and sulphur dioxide (SO2) its impact on the environment is significant and cannot be neglected. Emissions are so high that the European commission has formulated a document called Flightpath 2050, where it defines two of its goals to be reduction of CO2 and NOx emission by 75% per passenger kilometer and 90%, respectively. Currently, commercial aviation is responsible for around 3.5 % of the global carbon dioxide and nitrogen oxide emission [17] and it can be assumed that in an increasingly globalized world the demand for fast and convenient mobility will not diminish. Industry and governments around the world realized that taking active steps towards a cleaner flying are essential for saving the planet earth. In 2010, the international Civil Aviation Organization (*ICAO*) formulated industry-wide goals for reducing carbon emissions. These goals can be summarized by 2 main objectives, to establish carbon-neutral growth beyond year 2020 and to further reduce carbon emissions to half of the current level by the year 2050.

One simple way of reducing the overall carbon emissions already set to practice is the application of single-engine taxiing. With a single operating engine the fuel emissions are reduced only for the time in which the taxiing and waiting are carried out. Considering a whole flight mission, taxiing and waiting represents only a small fraction of the mission. Thus, this is not enough. Other techniques to reduce fuel consumption already in use are flying at an optimal cruise speed or using more environmental-friendly handbook trajectories. Methods which still acquire much exploration, however, can be assumed to play a main part in reducing

fuel consumption. These are overall new aircraft configurations, e.g. blended wing body and applying technologies like laminar flow control, boundary layer ingestion, ultra-high bypass ratio engines, more and all electric aircraft. The possibility to combine multiple technologies is given and is expected to lead to best results. The proposed *missioninformer* is not restricted to any aircraft configuration nor to any new applied technological advancements.

The demand for more environmental-friendly aircraft is becoming steadily louder. Furthermore, fuel price has increased and is forecasted to increase in the future. Among others, one of the main objectives of the *DLR*'s institute *Aerodynamics and Flow Technology* is to fulfill the described demands. Therefore, several optimization process were developed here to find structural, aerodynamical and engine optimized aircraft designs, details can be found in [13, 10]. The fuel consumption is targeted as the objective function inside an optimization process. The big advantage of such an optimization would result in reducing the CO2 emissions and thus pleasing the environmental demands. The *missioninformer* is written in a modular manner as mentioned earlier in this chapter. Therefore, it can be easily implemented into the *DLR*'s multi-disciplinary, -point and -objective optimization frameworks. Also, the user is enabled to define his own mission or multiple missions. Therefore, this feature could be used to enhance the already existing multi-point capabilities for the optimization. The importance of a multi-mission-optimization becomes visible, when thinking about an aircraft, which is used for a wide variety of missions, e.g. for different kinds of ranges, flight altitudes, speeds and payloads. In case of a single mission optimization, the given flight parameter will be suited optimally. However, a small change in these conditions could result into unbearable consequences of the fuel consumption. In practice, airlines want to buy aircrafts, which can be used for more than just one mission. A multi-mission optimized aircraft performs not as good for each condition, however it will cover a variety of missions and conditions within reason. The missioninformer, however, is not only suited for multi-point optimization, rather the enhanced multi-mission optimization. For instance in the previously mentioned processes masses for which the aircraft is trimmed are defined. Masses are defined through the method of multi-points, i.e. these are only frozen and discontinuous discrete aerodynamic states for which an optimization is carried out. In case of the multi-mission capable *missioninformer*, the masses are defined continuously, which comes closer to the reality of flying and thus consuming fuel. Furthermore, the *missioninformer* takes the snowball mass effect, which is described in section 2.4 into account. Also, since the input data is given by the aerodynamic output of *DLR*' analysis and optimization workflow *FSAerOpt* [25] involving the flow solver *TAU*, each mission fuel mass computation gets aerodynamical data for a fully trimmed state.

With this introduction the main 8 tasks for developing the *missionionformer* from scratch shall be mentioned:

1. Literature review on mission analysis and review of Breguet's range formula.
2. Familiarization with the gradient-based optimization process
3. Conception of an interpolation module for mission analysis
4. Implementation of the Python module
5. Derivation of the module according to the optimization parameters
6. Verification of the derivation in case the derivation is done by complex-step or analytically

7. Interface for a gradient-based optimization process

8. Identification of a best practice

As far it is possible, each task will be elaborated in this article. Since the tasks required much coding, it is important to mention the used programming language and the dependencies. As for the programming language, *Python 2* and *Python 3* were chosen. For the libraries *Matplotlib, Scipy, NumPy* and *DLR*'s own Python library *SMARTy* are deployed. For local coding a Linux workstation provided by the *DLR* with the following hardware configuration was used: *CPU:* Intel(R) Xeon(R) CPU E3-1270 v3 @3.50GHz and 32GB of RAM. Calculations were performed on the following high performance computer *CARA*: *CPU:* 2x AMD EPYC 7601 (32 cores; 2,2 GHz) per node and 2168 nodes with 128 GB DDR4 (2666 MHz) RAM. All calculations performed in this report used the *64* cores option. Some parts of *missioninformer* where NumPy is used, can be assumed to benefit from this option. However, the missioninformer itself is not parallized.

1.2 State of the art

The idea to reduce the entire fuel consumption is not new and a summary of the available methods is given in [6] and a more recent one in [15]. However, some works still shall be recited. Flight Optimization System (*FLOPS*) is a modular approach allowing the user to select appropriate modules for a given mission [23]. The aerodynamical data are generated by using the *Delta Method*, which is an empirical drag prediction method [9]. *FLOPS* enables some numerical optimization schemes, e.g. Davidon-Fletcher-Powell, Broyden-Fletcher-Goldfarb-Shano and a Quadratic Extended Interior Penalty method [15]. The gradients are calculated using finite differing, which is easy to apply, however an adequate derivation step size is required. The accuracy of gradients may not be sufficient and in case of a high number of design variables parallelization should be considered in order to get gradients in an acceptable time limit. Transport Aircraft System Optimization (*TASOPT*) is another well known tool for mission analysis and optimization [12]. The benefit of the latter is that physics-based models, rather than empirical data-tables are applied. Using data-tables results in interpolation could result into not realizable designs. The physics for *TASOPT* is obtained mostly by low-order models.

pyACDT is an aircraft conceptual design tool [26] containing a mission analysis component. It uses the Breguet range equation and empirically obtained fuel fractions for each flight mission segment. Liem et al. [19] take a similar approach, however with the advancement, that fuel fractions are only going to be used for the startup, taxi and landing segments of the mission. The fuel burn for climb, cruise and descent segments are calculated using a range equation and the flight equilibrium equations. The different segments are linked together such that the fuel weights at the end points of two neighboring segments must be equal. Another well known tool for aircraft conceptual design is developed by Lissys Ltd and is called *PIANO*. It is freely available, however, only for Windows and thus not compatible for HPC-based optimization processes.

Probably the two most well known mission analysis tools are *pyMission* and *SUAVE*. *pyMission* is developed by the authors of [15] and is now part of the openMDAO framework [11]. It uses the flight equilibrium equations and a fuel burn rate equation. In order to solve the fuel burn rate equation, which is an ODE, the amount of fuel carried at the end

of the mission is provided as the initial condition. The explicit Euler scheme starting from the end of the mission is applied. Thus, marching backwards in distance to the start of the mission is performed. For the rest of the mission fuel fractions are used. The gradients (total derivatives) for the optimization performed are obtained through the adjoint method [21]. *SUAVE* is a conceptual aircraft design framework developed at Stanford University and can be used for conventional and unconventional configurations. Tutorials as well as documentation for *SAVE*'s mission analysis tool can be found on their homepage [1]. In [18, 19, 20] the mission analysis is applied by using surrogate models and mixture of experts. Different kind of Kriging methods and RBFs (Radial Basis Function) are used to get interpolation models for lift, drag and pitching moment coefficients from a four-dimensional input space with variables, Mach number, angle of attack, flight altitude and tail rotation angle. For startup, taxi, takeoff and landing fuel fractions are used. The fuel burn for climb, cruise and descent segments are derived through numerical integration of a range equation. The results of the paper can be summarized as follows: the adaptive sampling improves the accuracy of surrogate models, the convergence was to be found slow in some cases, particularly when modeling complex profiles. Traditional surrogate models (RBF and Kriging without mixture of experts) performed well for the simpler lift and pitching moment coefficient (C_L, C_M) profiles. Using a mixture of gradient enhanced Kriging (GEK) models to approximate drag coefficients gave approximation errors of less than 5% with less than 150 samples, whereas the adaptive sampling failed to converge when training a global model. Since the traditional surrogate models approximate less complex behavior well, the authors of the papers recommend applying traditional surrogate modelling instead of mixture of experts. Latter requires much effort for its implementation. Finally, it is worth mentioning Dabas et al., who optimize the pylon shape by coupling an aerodynamic optimization with a mission analysis tool in order to propagate aerodynamic shape modifications to mission level [7]. The objective function, however is not directly the amount of fuel burnt, but rather the Cash Operating Costs (COC), which contain the burnt fuel as part of its weighted components.

1.3 Introducing missioninformer

In this section the overall processes and the functionalities of the proposed tool *missioninformer* shall be presented in an abstract way, without intern solved equations at this stage. It is a code written entirely in *Python* and makes use of the following libraries:

1. NumPy
2. SciPy
3. matplotlib
4. SMARTy (DLR's own Python library)
5. re (regex)

Its main workflow can be described with the figure 1.1. It obtains two input files and outputs the mass of total fuel which is required for the given mission and the gradients of the mission fuel with respect to shape parameters. The mission input file is completely written in *Python* and uses a regular dictionary, which is similar to a json file for storing purposes. In order to define one mission in the mission input file, the following 3 groups of parameters need to be set. The first group, named masses, contains weights:

1. maximal takeoff weight
2. maximal landing weight
3. operating empty weight
4. manufactures empty weights
5. maximum zero fuel weight
6. maximum fuel weight
7. design payload
8. maximum payload

Figure 1.1: Broad overview: Workflow *missioninformer*

The second group, named flying parameters, consists of the cruise Mach number (Ma), cruise altitude (h) and cruise range. The third group can be adjusted optionally by the user. It allows to change the step size for the central differencing scheme, which will be discussed in section 3.2 and define a weighting factor for each mission. The step size steers the accuracy and reliability of the calculated gradients. In case, multiple missions shall be used, for each mission a weighting factor must be given. Using multiple missions or adding missions is done easily by copying the section of the parameters and redefining these with desired values. Each new mission is summed to as one overall variable and needs to be passed as an argument to the mission collection method.

The second input for the *missioninformer* is a database. It is required to generate surrogate, i.e. interpolation models for the state variables $LoD, AoA\,(\alpha), TSFC$ and their gradients with respect to all shape parameters. LoD is called glide ratio, sometimes referred to performance and defined as: $LoD = \frac{C_L}{C_D} = \frac{L}{D}$, where the prefix C stands for coefficient and the letter L and D are denoted as lift and drag, respectively. AoA is the **a**ngle *o*f **a**ttack and is commonly denoted with the greek letter α. $TSFC$ is called the **T**hrust **S**pecific **F**uel **C**onsumption. In order to get an interpolation model for the state and gradient variables, the database must contain values for those and their associated inputs $Ma, h, mass$. The $mass$ is the total mass at the related condition for which the aircraft is trimmed. In simpler terms, considering figure 1.2, it can be observed that for combinations of $Ma, h, mass$ their associated known values for LoD are passed to an interpolation generator. Having obtained an interpolation model, for any given set of $Ma, h, mass$ the corresponding $\tilde{LoD}$ can be calculated. It should clearly stated that, the closer the input set ($Ma, h, mass$) values are to those which were used to generate the interpolation model, the better the prediction for the outcome $\tilde{LoD}, \tilde{AoA}, \tilde{TSFC}$ is. This means, if the given input set is highly out of the range of the training data, then an absurd extrapolation may occur. Figure 1.2 only depicts the workflow for obtaining an interpolation model for LoD. However, analogously to the explained proceeding by only replacing LoD with the desired output variable ($AoA, TSFC$ and their gradients), 3 interpolation models for the state variables and the remaining for the gradients can be achieved.

Since for each state variable ($LoD, AoA, TSFC$) the gradient with respect to each shape parameter is required, the number of the interpolation models for the gradients is directly connected to the number of the shape parameters. For example, having 126 shape parame-

ters, results in having 126 gradients for each state variable. Thus, 126 interpolation models for the gradients of each state variable (LoD, AoA, $TSFC$) is necessary. In total, $126 * 3 = 378$ interpolation models only for the gradients are required.

Figure 1.2: Exemplary interpolation model generation

Even though it might seem to be a heavy computation it is much faster compared to the CFD calculation performed for each sample point to provide the state variables and their gradients as training data. By using the interpolation approach the integration of the Breguet's range equation and the iteration of the mission fuel consumption becomes feasible although using accurate but time-expensive CFD-based data. Furthermore, the tool scales nicely with an increasing number of missions. However, the disadvantage is that the quality of the calculations completely depends on the quality of the interpolation models. The interpolation model itself depends on the accuracy of the provided input or training data as well as the underlying method. Furthermore, the final goal is to use the *missioninformer* within a gradient based optimization. The optimizing algorithm then heavily relies on the correctness of the gradients. Additionally, gradient computations itself is a sensitive calculation to do. With the importance of the correct chosen interpolation model in mind, in section 2.5 two commonly used interpolation methods and several options are investigated.

The database containing the training samples is given by the output of *DLR*'s aerodynamic analysis and optimization workflow *FSAerOpt* [25] involving the flow solver *TAU*. It is calculated at trimmed states, however, trimming is not always feasible. In this case, *missioninformer* automatically identifies the not trimmed values and does not include them into the intern *Python* database. This feature is part of the modular skills of the missioninformer. It thus enables an easy extension or insertion to an already existing workflow. This is reinforced by the type of output, i.e. the total fuel burn and their gradients are stored as simple ASCII text files and as NumPy arrays as well. In case of a *Python* based workflow, the NumPy arrays can be loaded into the RAM (Random Access Memory) with one additional line of code.

1.3.1 Workflow

A more detailed workflow is depicted in figure 1.3. Before discussing the figure, note that, this is still a broad workflow, and it is supposed to provide only a general and not a deep understanding. Among other, the cruise segment fuel equation, its derivative, its solution, the used interpolation models and outer loops for example for the mass recalculations are skipped for now. The detailed elaboration happens in chapter 2. The workflow 1.3 can be read as follows: After having generated the database containing values for LoD, AoA, $TSFC$ and their gradients for a given set of inputs (Ma, h, $mass$), the mission input file can be adjusted in order to define the flight mission. In case of a multi-mission definition one more step regarding weighting is done, which is not depicted in the current workflow overview. With

these inputs *missioninformer* generates interpolation models for states and for gradients. In order to get the mass of cruise segment fuel, its equation is solved. This equation requires the state surrogate models to be called. In fact, the solving of state cruise segment fuel equation is numerical and iterative and thus requires multiple function calls. In other words, solving the cruise segment fuel equation once, invokes the surrogate models multiple times. The gradients of the cruise segment fuel mass needs the state and the gradient interpolation models and also invokes both models multiple times for one solution. Contrary to the generation of the interpolation models, the invocation of the already calculated interpolation model is not computationally costly. As mentioned, a deeper insight into the whole process is provided in chapter 2. Once the state and gradients of the cruise segment fuel mass are calculated they are written to hard disk as output.

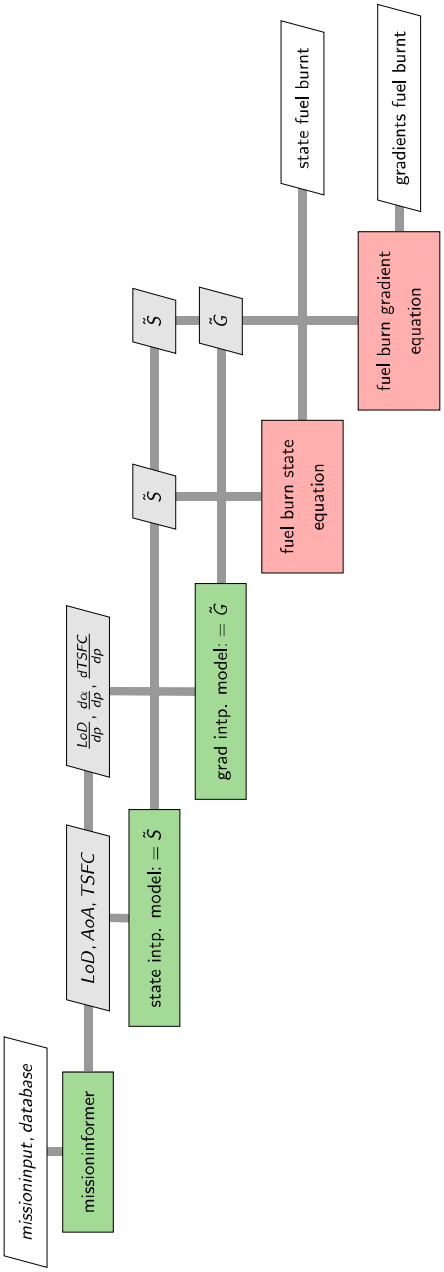

Figure 1.3: Second workflow overview with some more details

2 Methodology

In this chapter the employed equations, iterations and surrogate models for obtaining the state and gradients of the burnt fuel and their theoretical background shall be given. Additionally, the way of exploring methods, the reasons for having chosen the methods and the proceeding to obtain solutions shall be explained. Furthermore, some workaround which was required will be mentioned, e.g. in the upcoming section, where a small tool was developed in order to get the atmospherical relationships.

2.1 Atmospheric conversions

One term inside the cruise fuel burn equation for the computation of the cruise segment fuel is the speed of sound (a), which is not given by the user. However, what the user provides is the altitude. By making use of one of the 3 common atmospheric models, ICAO-, US-Standard Atmosphere or Norm Atmosphere DIN 5450/LN 9300, a relationship between pressure, density and altitude (p, ρ, h), can be obtained. For our purposes, the *ICAO-Standard Atmosphere(ISA)* and *Norm atmosphere DIN 5450/LN 9300* were used within an altitude range of 0-20 km (troposphere, lower stratosphere) and 20-32 km (upper stratosphere), respectively. With the mentioned atmospheric models, altitude can be obtained by providing pressure and density and vice versa. The *missioninformer* is capable of both and by following the equation (2.1) the speed of sound (a) can be calculated. The *heat capacity ratio* is chosen to be $\kappa = 1.4$. The reviewed process is depicted in figure 2.1

$$a = \sqrt{\kappa \frac{p}{\rho}} \tag{2.1}$$

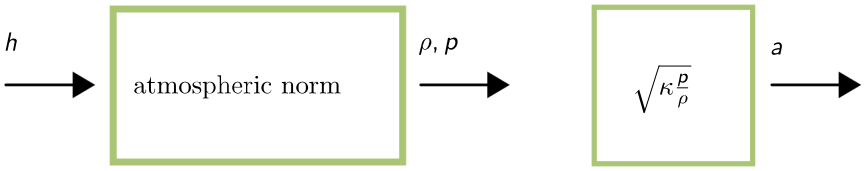

Figure 2.1: Atmospheric model and calculation of the speed of sound a

2.2 Cruise and mission fuel mass

In this section the derivation of the fuel burn equation for the cruise segment and the applied solution method shall be explained. Since the *missioninformer* considers the whole mission it will also be elaborated how the fuel masses for the remaining flight segments are obtained.

The derivation of the cruise segment fuel burn equation, which takes the form of an Ordinary Differential Equation (ODE) was provided by Ilic in an *DLR*-intern report [14]. For

its derivation a constant altitude ()$h = const.$) is assumed. The cruise fuel burnt mass is the mass of fuel, which is required for the airplane in order to reach or fly a given range. The ODE is given in equation (2.2), where m_{fe} is the burnt fuel mass, called fuel expended accordingly to the original report [14]. The derivation is applied with respect to the range ds, $g, a, Ma, TSFC$ are denoted as the gravitational constant, the speed of sound, the Mach number, and $TSFC$ thrust specific fuel consumption, respectively.

$$\frac{dm_{fe}}{ds} = \frac{g}{a}\,\frac{1}{Ma}\,TSFC\,(m_s - m_{fe})\,\frac{\frac{C_L}{C_D}\,tan(\theta) + 1}{\frac{C_L}{C_D}\,cos(AoA) + sin(AoA)} \tag{2.2}$$

The fraction $\frac{C_L}{C_D} = \frac{L}{D} = LoD$ is called aerodynamical performance or glide ratio. AoA, θ, m_s are denoted as the angle of attack, flight path angle and cruise starting mass. The presented equation is only one possible equation formulation, which can be employed to find the cruise fuel burnt mass. In order to understand why this is so, its derivation shall be explained briefly. In his report [14], the author starts with the equilibrium equations for steady flight. By inserting them into each other and using equation transformations Ilic [14] comes to an expression for the thrust as: $T = f(m, g, LoD, AoA, \theta)$. At this stage, the change of the fuel mass with the distance traveled can be expressed in the following two ways:

$$\frac{dm_{fr}}{ds} = -\frac{TSFC}{Ma\,a\,cos(\theta)}\,T \tag{2.3}$$

$$\frac{dm_{fe}}{ds} = \frac{TSFC}{Ma\,a\,cos(\theta)}\,T \tag{2.4}$$

Because the present total airplane mass m in the thrust expression $T = f(m, g, LoD, AoA, \theta)$ can be expressed in two ways, two equations (ODEs) are possible as outcome to describe the cruise fuel mass burnt. One is over the remaining fuel mass m_{fr}

$$m = m_e + m_{fr}, \tag{2.5}$$

where m_e is the end mass or the mass right before landing. The second ODE is obtained by using the expended fuel mass as:

$$m = m_s - m_{fe}, \tag{2.6}$$

where m_s is the cruise start mass or the mass just before cruise, after climb and acceleration. According to common field specific knowledge, masses in aerospace can be further distributed into detailed definitions, e.g. the takeoff mass can be further split up in operating empty weight, payload and fuel weight. The operating empty weight itself can be split up into airframe structure, propulsion group, airframe services and equipment, fixes equipment, removable equipment, standard items, standard item variation and operational items [30]. Depending on the field of research and the desired level on accuracy the mass division can be continued. However, the start mass m_s from equation (2.6) contains the fuel weight for all segments which are followed after climb and acceleration, cruise till taxi to parking. During the flight this fuel mass is going to be reduced. With this background the two upcoming equations can be looked at:

$$\frac{dm_{fr}}{ds} = -\frac{g}{a}\,\frac{1}{Ma}\,TSFC\,(m_e + m_{fr})\,\frac{\frac{C_L}{C_D}\,tan(\theta) + 1}{\frac{C_L}{C_D}\,cos(AoA) + sin(AoA)} \tag{2.7}$$

$$\frac{dm_{fe}}{ds} = \frac{g}{a}\frac{1}{Ma}\,TSFC\,(m_s - m_{fe})\,\frac{\frac{C_L}{C_D}\,tan(\theta) + 1}{\frac{C_L}{C_D}\,cos(AoA) + sin(AoA)} \tag{2.8}$$

The difference between both comes through their derivation and thus the thought process or undertaken assumptions. Equation (2.7) works with remaining fuel, the end mass m_e is known, the fuel is calculated and with these the start mass m_s is obtained, see equation (2.5). This version lets the start mass be variable and requires the end mass as input. For equation (2.8) the start mass m_s is passed as an input argument, the expended or required fuel for given cruise range is calculated.

Comparing the solution methods for the ODEs (2.7) and (2.8), the first is integrated backwards from cruise range end to zero. In other words, the initial condition is given with the assumption that the remaining fuel at the end of the range equals zero $m_{fr}(R = R_{cr,max}) = 0$, where R is denoted as the range. As mentioned in section 1.2, *pyMission* follows this approach for solving the ODE. In contrast, for defining the initial condition for equation (2.8) the fuel mass at the cruise start or cruise range zero is provided. It can be understood, that before the cruise even starts, no fuel in the cruise section is burnt. Note, the presented ODEs are only valid for a cruise segment. Therefore, just at the start of the cruise segment, neglecting other flight segments for this consideration, no fuel within the cruise segment is burnt. Thus, the initial condition is formulated as equation (2.9). In this way, the ODE is integrated forward from 0 to given range.

$$m_{fe}(R_{cr} = 0) = 0 \tag{2.9}$$

For this work the viewpoint of equation 2.8 considering fuel expended or burnt is going to be applied in order to derive the upcoming solutions. Furthermore, the flight path angle is set to zero $\theta = 0$, see equation (2.10). Additionally, with the introduced relationship $LoD = \frac{C_L}{C_D}$ the final ODE can be formulated as equation (2.11)

$$\frac{dm_{fe}}{ds} = \frac{g}{a}\frac{1}{Ma}\,TSFC\,(m_s - m_{fe})\,\frac{1}{\frac{C_L}{C_D}\,cos(AoA) + sin(AoA)} \tag{2.10}$$

$$\frac{dm_{fe}}{ds} = \frac{g}{a}\frac{1}{Ma}\,TSFC\,(m_s - m_{fe})\,\frac{1}{LoD\,cos(AoA) + sin(AoA)} \tag{2.11}$$

Before explaining the methodology for solving the chosen ODE (2.11), the inclusion of the remaining flight segments shall be explained. In [27] fuel fractions, which are the ratio of the aircraft total weight at the end of a flight segment to the weight at the start of the same segment, were provided. The employed fuel fractions for the associated flight segment are presented in the table 2.1. The obvious advantage of applying empirical based fractions instead of solving physical based equations for flight segments other than cruise, is the absence of time and thus money consuming modeling and solving of physical based equations and as consequence a fast as well as easy implementation. Also, since only a multiplication is performed, the output in *Python* is obtained within more than a feasible time. The disadvantage however, is a loss of accuracy. Given the fact that most fuel is burnt during the cruise segment, this seems a valid approach. For understanding, why the reserve fuel fraction is set to 0.05, consider the equation (2.69) in the subsection 2.6.1.

Flight segment	Fuel fraction
engine start	0.99
taxi to runway	0.99
takeoff	0.995
climb and acceleration	0.98
descent	0.99
landing	0.992
taxi to parking	0.99
reserve	0.05

Table 2.1: Fuel fractions for non cruise flight segments

2.3 Method for solving the ODE

The chosen ODE (2.11) can be solved numerically with different kind of solvers. However, in order to have a possibility to validate the quality of the numerical solutions, some simplifications shall be introduced. With these, it is possible to derive an analytical solution. The variables $LoD, AoA, TSFC, m_{fe}$ are all functions of the present total mass, which is shown in equation (2.12). Since the start point for the ODE to be valid is at cruise, the current total mass is the cruise fuel starting mass $m = m_s$. However, itself has a dependency on the expended fuel mass, which will be elaborated in section 2.4 and makes an analytical explicit solution hard to be derived. Additionally, no equations for $LoD(m_s)$, $AoA(m_s)$, $TSFC(m_s)$, which would describe a relationship between the variable and the total present mass, are given and therefore formulating an analytical solution becomes impossible.

$$\frac{dm_{fe(m)}}{ds} = \frac{g}{a}\frac{1}{Ma}\,TSFC(m)\,(m_s + m_{fe}(m))\,\frac{1}{LoD(m)\,cos(AoA(m)) + sin(AoA(m))} \tag{2.12}$$

At this point the mentioned simplifications for an analytical solution can be mentioned. Assuming $LoD, AoA, TSFC$ to be constant allows to find an analytical solution, as is presented in the upcoming equations. Consider the equation (2.13), where the A and B are constants and x is the variable for which the solution it is pursued.

$$\frac{dx}{ds} + A\,x = B \tag{2.13}$$

The solution to this problem (2.13) can be obtained by using the integrating factor given in (2.14), where C is a new constant. By replacing $x = m_{fe}$ equation (2.15) is obtained.

$$x = \frac{B}{A} + C\,e^{-A\,s} \tag{2.14}$$

$$m_{fe} = \frac{B}{A} + C\,e^{-A\,s} \tag{2.15}$$

It is understood that, the consumed fuel mass at the start or at cruise range zero has to be zero $m_{fe}(R_{cr} = 0) = 0$, which is the initial condition to the equation (2.15). Inserting this observation leads to equation (2.16)

$$\begin{aligned} m_{fe}(s = R_{cr} = 0) = 0 &= \frac{B}{A} + C \\ \Rightarrow C &= -\frac{B}{A} \end{aligned} \tag{2.16}$$

Defining A and B as equations (2.17) and (2.18), respectively and incorporating equation (2.16), the analytical solution for the cruise fuel burn can be written as equation (2.19).

$$A = \frac{g}{a}\,\frac{1}{Ma}\,TSFC\,\frac{1}{LoD\,cos(AoA) + sin(AoA)} \tag{2.17}$$

$$B = A\,m_s \tag{2.18}$$

$$\begin{aligned} m_{fe}(s = R_{cr} = 0) &= \frac{B}{A} + \frac{-B}{A}\,e^{-A\,s} \\ \Leftrightarrow m_{fe} &= \frac{B}{A}(1 - e^{-A\,s}) \\ \Leftrightarrow m_{fe} &= \frac{B}{A}(1 - e^{-A\,s}) \\ \Leftrightarrow m_{fe} &= \frac{A\,m_s}{A}(1 - e^{-A\,s}) \\ \Rightarrow m_{fe}(s = R_{cr}) &= m_s\left(1 - e^{\frac{-g}{a}\,\frac{1}{Ma}\,TSFC\,\frac{1}{LoD\,cos(AoA)+sin(AoA)}\,s}\right) \end{aligned} \tag{2.19}$$

This analytical expression in equation (2.19) is compared with the output of different numerical solvers of Pythons open source library *SciPy*. The different numerical ODE solvers are within the method *solving initial value problem (solve_ivp)* and each solver comes with several options, which are documented in [4]. The results and the comparison is given in section 3.1. However, the conclusion of this investigation is that, *SciPy*'s Runge Kutta of order 5 (*RK45*) can be employed without any noticeable loss of accuracy. *RK45* assumes an accuracy of the fourth-order method, but steps are taken using the fifth-order accurate formula. The occurring deviations to the analytically obtained solution (2.19) are less than milligrams, which exhibits a sufficient level of accuracy.

SciPy allows the user to define the number and the location of steps. However, having compared its default option with less than 20 steps to a ridiculous high number of 1460597 steps, no noticeable deviation could be observed. Fewer steps at which calculations happen results into a faster execution time. This becomes even more clear when recalling that each step invokes surrogate evaluations. In case of 20 used steps undertaken, in order to solve the cruise fuel consumption ODE, for each step in this iterative process, the corresponding values for $LoD, AoA, TSFC$ with the input set of $Ma, h, mass$ are necessary. In simpler terms, each surrogate model obtain an input set of $Ma, h, mass$ and provides the corresponding output for $LoD, AoA, TSFC$, which are predictions, thus $\tilde{LoD}, \tilde{AoA}, \tilde{TSFC}$. Since, this has to done be for each step, $20 * 3 = 60$ surrogate model invocations for solving the cruise fuel burn equation once, are required. Therefore, the low number of steps and because of the sufficient accuracy of the solution with *SciPy*'s default settings for *RK45* are chosen to solve the cruise fuel burn mass.

As a final recap, the cruise fuel burn equation cannot be solved analytically due to explained reasons and thus is solved with an outstanding accuracy employing *SciPy*'s numerical ODE solver *RK45* within the *solve_ivp* environment applying the default options.

2.4 Mission fuel mass iteration

For solving the cruise fuel burn equation (2.11) the present total mass of the airplane m_s, which is the starting mass for the cruise segment, is required. However, the cruise starting mass m_s is not known beforehand. It depends on the fuel mass of the remaining segments fuel masses. These are the fuel weights of descent, landing and taxi to parking. As explained in section 2.2, only the cruise fuel weight m_{fe} for the given mission or missions is calculated by solving the ODE (2.11). In this section m_{fe} is denoted as m_{cr} in order to highlight that it is only the fuel mass for the cruise segment. The required fuel masses for the remaining flight segments are obtained through empirical fuel fractions from table 2.1. The process for computing the total fuel mission mass m_f, the cruise fuel m_{cr} and the resulting cruise starting mass m_s can be viewed in the figure 2.2.

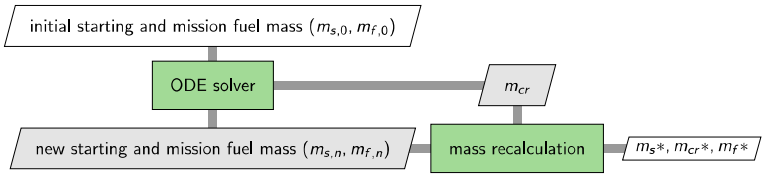

Figure 2.2: Initial mass and mass fix-point iteration workflow

Since the cruise starting mass m_s is not known before knowing the cruise and total fuel masses (m_{cr}, m_f), the initial cruise starting and total fuel mass ($m_{s,0}$, $m_{f,0}$) are estimated based on the equations (2.20) to (2.22). The variables for the equations $m_{takeoff,max}$, f_{zero}, $f_{eng,start}$ m_{oe}, $m_{payload}$ are denoted as maximal takeoff, zero fuel, engine start, operating empty weight and design payload, respectively. The fuel fractions fr_1 , fr_2 , fr_3 , fr_4 are for the following flight segments, after engine start, taxi to runway, takeoff and climb and acceleration, respectively.

$$f_{zero} = m_{oe} + m_{payload} \tag{2.20}$$

$$m_{f,0} = m_{takeoff,max} - f_{zero} \tag{2.21}$$

$$m_{s,0} = (f_{zero} + f_{eng,start})\ fr_1\, fr_2\, fr_3\, fr_4 \tag{2.22}$$

By proving $m_{s,0}$ to the cruise fuel ODE (2.11) the corresponding cruise fuel m_{cr} is obtained, which is passed to the block *mass recalculation*. Here the new cruise starting and total fuel masses ($m_{s,n}$, $m_{f,n}$) are calculated. For computing former ($m_{s,n}$) the equation (2.22) is still valid, however, computing the new total fuel mass $m_{f,n}$ two different possibilities were found. One by using all known mass relationships straight forward and solving $m_{f,n}$ numerically, e.g. the Newtons method. The second method was obtained by inserting the mass equations into each and performing transformations. With that an analytical solution was obtained. Here, only the derivation of the analytical solution shall be explained.

The demand for $m_{f,n}$ is stated in equation (2.24), where $m_{after,cr}$ is the mass after the cruise segment and fr_{cr} is introduced as auxiliary fuel fraction for the cruise segment and is given in equation (2.23). In words, the mass after the cruise segment divided by the mass after acceleration and climb must be equal to the cruise fuel fraction.

$$fr_{cr} = \frac{m_s - f_{cr}}{m_s} \tag{2.23}$$

$$\frac{m_{after,cr}}{m_s} - fr_{cr} \stackrel{!}{=} 0 \tag{2.24}$$

The mass after the cruise segment $m_{after,cr}$ can be written in the form of equation (2.25), where fr_{reser} is the known fuel reserve fraction, provided in table 2.1. The fuel fractions fr_5 , fr_6 , fr_7 stand for taxi to parking, landing and descent, respectively and are also given in table 2.1. The unknown in this equation is the new total fuel mass $m_{f,n}$. Applying insertions and mathematical transformations with the equations (2.22) to (2.25), $m_{f,n}$ can be written as equation (2.27), where fa as an auxiliary term is given in equation (2.26).

$$m_{after,cr} = \frac{f_{zero} + m_{f,n}\, fr_{reser}}{fr_5\, fr_6\, fr_7} = 0 \tag{2.25}$$

$$fa = f_{zero}\, fr_{cr}\, fr_1\, fr_2\, fr_3\, fr_4\, fr_5\, fr_6\, fr_7 \tag{2.26}$$

$$m_{f,n} = \frac{fa - f_{zero}}{f_{reser} - fr_{cr}\, fr_1\, fr_2\, fr_3\, fr_4\, fr_5\, fr_6\, fr_7} \tag{2.27}$$

To recap, after an initial guess for starting cruise and total fuel mass $m_{s,0}, m_{f,0}$, they are passed to the ODE solver, which outputs its corresponding cruise fuel weight m_{cr}. With the equation (2.23) an auxiliary fuel fraction for the cruise segment is calculated and used in the *mass recalculation* box, depicted in figure 2.2. Here by employing the equations (2.22) and (2.27) the new cruse starting and total fuel mass $m_{s,n}, m_{f,n}$ are obtained, respectively. This process is repeated as long the convergence criteria $\Delta = m_{f,n} - m_{f,old} \leq \epsilon$ is met, where ϵ is a tolerance parameter. The relative and absolute tolerance parameter are set to be $5e{-}5$ [2]. Such an iterative method is also called *fix-point iteration*. Since the described workflow is iterative, the resulting change and their interdependencies (mass snowball effect) are taken into account. In case a convergence is achieved, the final values for the cruise starting, cruise fuel and total fuel mass 2.2 are found ($m_s*, m_{cr}*, m_f*$).

As mentioned above, the problem (2.24) can be solved numerically or analytically. In order to verify the analytical solution it has been compared successfully with the numerical Newtons' method. The presented workflow defining the auxiliary term fr_{cr} (2.23) can be further simplified by directly inserting fr_{cr} into the equations. This results into equation (2.76) and its derivation with respect to the shape parameter is explained in more details in section 2.6. The results of this direct (without auxiliary term) approach differed marginally from the previous auxiliary method. Therefore, *Matlab* and *SymPy* (open source Python library) were used in order to verify the undertaken mathematical transformations. The symbolic results confirmed the handcrafted analytical solution.

One interesting finding could be observed. Consider the figure 2.2, which shows the reoccurring or iterative process, thus the fix-point iteration. Depending on applying the auxiliary or the direct version, different numbers of iterations were needed to get to the final solutions for $m_s*, m_{cr}*, m_f*$. As mentioned, also the end values for $m_s*, m_{cr}*, m_f*$ would differ, however only around the 10th decimal place. By changing the missions definitions, it could be observed that in most cases the direct version required more iterations to converge. In some cases, more than two to four times more iterations were required. Thus, *missioninformer's*

default method to solve equation (2.24) is by employing the auxiliary term fr_{cr}.

Finally, the *missioninformer's constraint violation check* shall be elaborated. Since the user is completely free in defining the desired mission or missions, it is possible for an unpracticed user to define an unrealistic high range or payload. Also, possible is that, e.g. the predicted $\tilde{LoD}$ with the preliminary aircraft synthesis software turned out to be too optimistic and at the detailed Reynold-Averaged-Navier-Stokes-Equations- based (RANS) aerodynamics analysis. In such cases the required fuel masses or other weights can exceed the mass constraints set by the top level aircraft requirements (TLAR) and the preliminary design. Therefore, verifying whether given constrains are violated is necessary. The workflow with the *constraint violation check* is depicted in figure 2.3. As it can be observed, it is performed after each mass recalculation. In case of a found constraint violation, a colored warning

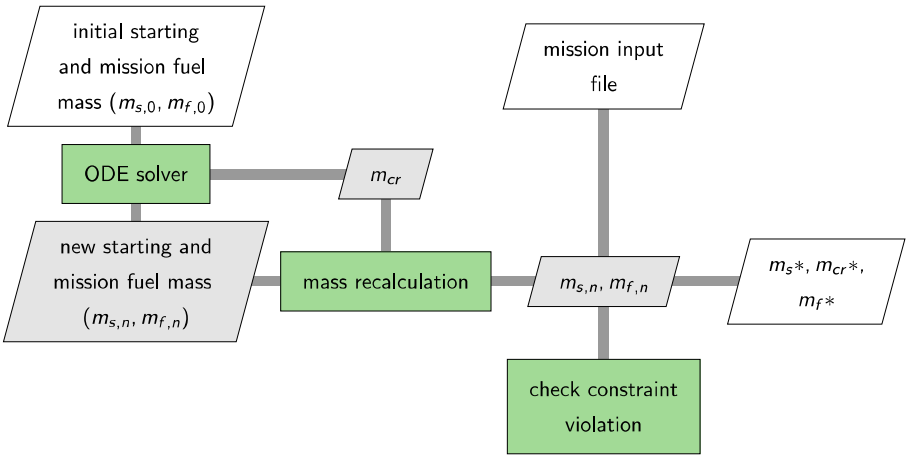

Figure 2.3: Second workflow with some more details

statement will be given to highlight an eventual inconsistency with the possible incorrectness of the mission inputs definition. The following checks are made, where $m_{takeoff,max}$, $m_{eng,start}$, $m_{land,max}$, $m_{aft,desc}$, $m_{f,max}$, m_f are denoted as the weights of maximal takeoff, engine start, maximal landing, after descent, maximal fuel and actual required total fuel, respectively.

$$\begin{aligned} m_{takeoff,max} &< m_{eng,start} \\ m_{land,max} &< m_{aft,desc} \\ m_{f,max} &< m_f \end{aligned} \tag{2.28}$$

2.5 Surrogate models

In very brief words, a surrogate model can be described as an approximation model. In most cases it is used in order to pack computational heavy physical based calculations into a simpler and execution time friendlier calculation environment. Surrogate models are very popular in the field of optimization, because every optimization algorithm evaluates function values with variation of the design variables. The design variables are the parameters which are changed by the optimizing algorithm in order to fulfil the minimization or maximization task. The design variables could consist of only one variable, e.g. the wing aspect ratio.

Another example for many design variables is topology optimization, where the number of design variables is equivalent to the number of the discretized finite elements. For a real world application, e.g. an aircraft or a car the number of finite elements can easily overreach millions. The optimization algorithm tries to minimize the objective function by changing the design variables and usually, each change demands a new calculation. With the variation of the design variables, the optimization algorithms collects outputs which are compared to each other in order to follow a steadily improving path.

In case of a flow field computation, changing only one parameter, e.g. the angle of attack (AoA), obligates a new expensive computational task. Depending on the applied fidelity and the object (airplane, car, turbo machinery) for which the flow field is solved, it could acquire seconds till years (Direct Numerical Simulation). Thus, one objective of surrogate models is enabling feasible computational times. Additionally, surrogate models can be generated and invoked on none high performance computers.

Surrogate models are also known as metamodels or models of models. Eldred er al. [8] classified surrogate models into three categories: data-fits, reduced order models and hierarchical models. Reduced-order and hierarchical surrogate models can be further classified as physics-based approaches, due to the exploitation and simplification of governing equations [5]. Therefore, these models are considered as *intrusive* methods. Data-fit surrogate models, however, are assigned to the *black-box* approach, where the outputs are only based on the inputs, without necessarily knowing the underlying equations. Black-box approaches are non-intrusive methods and can be further categorized into regression and interpolation. The main difference between these are, that interpolation must match the training or reference data at the given reference points, whereas regression is not obligated to match the sample data at the reference locations. Two widely used interpolation surrogate models are Radial Basis Function (RBF) and Kriging models. To generate and use the black-box intrusive surrogate models, sample data is required which for the *missioninformer* is provided by the output of the DLR aerodynamic analysis and optimization workflow *FSAerOpt* [25] involving *DLR*'s flow solver *TAU*.

After having the input or training data, a surrogate model and its intern tuning parameter with which the model shall be constructed needs to be defined. The generation of the surrogate model (offline phase) can claim some noticeable time, however, still in feasible manner. Invoking the already build surrogate model (online phase) can be compared with regard to execution time to solving a simple linear equation, which is satisfactorily feasible. The main disadvantage by applying surrogate models is the loss of accuracy compared to the underlying physics based equations or the measured experimental data. For assessing the quality of the surrogate model some methods are presented in [24]. However, in *missioninformer* one of the most commonly used error measurement, the *root mean square error* (RMSE), is employed. It is given in equation (2.29), where n, y_i, $\tilde{y}_i$ are denoted as the number of validation points (training or sample data obtained from workflow of *FSAerOpt* [25]), the actual value and the predicted value, respectively.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2} \tag{2.29}$$

The *missioninformer* involves two interpolation surrogate models, Kriging and RBF with different options, which are described in section 2.5.1 and 2.5.2. Both models will be tested with respect to execution time, robustness and RMSE with different model specific options in section 3.4. Surrogate models within the *missioninformer* are generated for $\tilde{LoD}$, $\tilde{AoA}$, $\tilde{TSFC}$ and their gradients with respect to shape parameter. The input set for the mentioned desired output state and gradient variables consists of Ma, h, m_s. The state surrogate models are invoked for solving the equation (2.11), thus needed for the cruise fuel mass computation. For the rest of the flight segments no surrogate models are required, since fuel fractions are exploited as explained in sections 2.2, 2.3 and 2.4. Solving the state fuel cruise mass invokes the three required surrogate models to predict $\tilde{LoD}$, $\tilde{AoA}$, $\tilde{TSFC}$ at each step undertaken by the chosen *RK45* solver [4]. The total number of state surrogate model invocations for the state can be described as equation (2.30). The number of the gradient surrogate model invocations further includes the number of the shape parameter. The required number of gradient surrogate models and their number of invocations are given in equations (2.31) and (2.32), respectively.

$$\#_{invok,state} = 3 \cdot steps_{RK45} \tag{2.30}$$

$$\#_{gener,grad} = 3 \cdot \#_{shape,param} \tag{2.31}$$

$$\begin{aligned} \#_{invok,grad} &= 3 \cdot steps_{RK45} \cdot \#_{shape,param} \cdot 2 \cdot \#_{fix,pkt,iter} \\ &\Rightarrow 6 \cdot steps_{RK45} \cdot \#_{shape,param} \cdot \#_{fix,pkt,iter} \end{aligned} \tag{2.32}$$

In order to understand why equation (2.32), briefly missioninfomers implemented method to derive gradients shall be mentioned, which in detail is explained in subsection 2.6.2. The gradients in *missioninformer* are obtained by exploiting the Central Differencing Scheme, which basically adds a very small positive and negative perturbation to the current state of the present desired variable ($LoD, AoA, TSFC$), for which the gradient is desired and divides the result by two times the size of the perturbation. Due to the results for both perturbed states, positive and negative added perturbation, the number of surrogate model invocation increases. In simpler terms, the surrogate model is invoked once for positive added permutation and once for negative added perturbation. The perturbation is called step-size. How and why *missioninformers* default step-size has the value $1e{-}5$ is described in subsection 2.6.2 and section 3.2.

Additionally, also the final values for the gradients are obtained after a fix-point iteration. In cases of a barely solvable mission more than 500 iterations were required, in some cases even no solution was obtainable and finally in cases where the missions were defined properly (not unrealistic high ranges or payloads) only 4 fix-point iterations were required. As for the tested missions, a correlation could be read out. In cases where the constraint violation warning was given, more than 4 fix-point iterations could be seen. Therefore, it is advised to observe the warning outputs. Note, the number of the fix-point iterations is also depended on the chosen step-size. The number of 4 fix-point iteration is met due to the well-fitted step-size of $1e{-}5$. The step-size can be changed inside the mission input file for each mission. However, the default value in *missioninformer* is set to $1e{-}5$ and a change in its value is assumed to most likely have a negative impact on the number of fix-point iterations. The reasons for that will become clearer by considering the convergence behavior depicted in the

step-size study in section 3.2.

Assuming having 126 shape parameter, 20 required steps for the ODE's integration and 4 fix-point iterations, the gradient surrogate models are invoked 30240 times. Using directly CFD calculations, where one CFD result takes easily 3 hours using a reasonable amount of high performance computer resources.

Finally, it shall be mentioned that the number of fix-point iterations is limited to 500 iterations. In cases, where the missions are physically correctly defined and no constraint violation warning were observed, not more than 5 fix-point iterations were encountered. Therefore, it is regarded to be highly unlikely to require more than 500 iterations. When the limit of 500 iterations is reached, the user will be notified in the output about the abortion due to the exceeding of the fix-point iteration number's limit.

2.5.1 Kriging models

Kriging for using as a surrogate model was originally developed for the field of geostatics by Daniel G. Krige, after which the method is named [16]. The term *Kriging* was shaped by Matheron [22], who was also the first to formulate it mathematically. Initially after its development, Kriging was used to model continuous defined functions based on measurement data. The foundation of exploiting Kriging models in the Design and Analysis of Computer Experiments (DACE) was first developed by Sacks et al. [28], where the input points represent the spatial geographical coordinates. In Kriging models a deterministic response $y(x)$ is a realization of a stochastic process $Y(x)$ [28] and is described by the following equation:

$$Y(x) = \sum_{k=1}^{N_f} f_k(x)\beta_k + Z(x) = f^T(x)\beta + Z(x) \tag{2.33}$$

The first term is the global model component, where $f(x) = [f_1(x),\ f_2(x),\ ....,\ f_{N_f}(x)]^T$ is a vector of N_f basis functions and $\beta = [\beta_1,\ \beta_2,\ ...,\ \beta_{N_f}]$ is a vector of the unknown coefficients. The stochastic component $Z(x)$ is treated as the realization of a stationary Gaussian function with an expected value of zero ($E[Z(x)] = 0$) and the covariance is given in equation (2.34), where R denotes the correlation function with $R(0) = 1$. Because of this, Kriging models can be used as interpolation models and thus provide the exact prediction at the training or sample data. However, by applying regularization, which adds a constant to each control point, the interpolation turns into regression. The greater the distance of the input, for which the predicted output is desired is from the training sample data, the greater the error in the variance gets. In other words, in Kriging models the data are assumed to be exact, but the function is a realization of a Gaussian process [31]. The second term is denoted as bias, a systematic departure from the linear model or localized deviation [29].

$$Cov\left[Z(x_i),\ Z(x_j)\right] = \sigma^2\, R(x_i, x_j) \tag{2.34}$$

In *1d* the stationary correlation function between any two points in the input space $y(x_i)$ and $y(x_j)$, depend only on the difference between these points ($\Delta x = x_i - x_j$). For higher-dimensional problems, the correlation function in a Kriging model usually obeys the *product correlation rule*, given in equation (2.35), where $\theta = \theta^{(k)},\ k = 1, ..., N_d$ is denoted as the vector of correlation parameters. The notation $d_{ij}^{(k)}$ is the distance between two points in the k^{th} dimension ($x_i^{(k)} - x_j^{(k)}$). The correlation parameters, are called Kriging hyperparameters and can be found by using the Maximum Likelihood Estimation (MLE) approach. Large

θ values represent a weak spatial correlation, whereas small values stand for a strong spatial correlation. Since *missioninformer* calls *SMARTy* for its Kriging operations, different correlations functions can be used. However, within this research the Gaussian correlation function is used.

$$R_{ij}(\theta, d) = \prod_{k=1}^{N_d} R(\theta^{(k)}, d_{ij}^{(k)}) \tag{2.35}$$

2.5.2 RBF models

RBF stands for *Radial Basis Function* and is black-box surrogate model, which simulates complicated design landscapes using a weighted sum of simple functions as shown in equation (2.36). Here x_0, x_s, y_s are denoted as input space, samples location's and output's value, respectively. The function $\psi(\,||x_0 - c_i||\,)$ is the kernel function centered at c_i and the norm $||\cdot||$ is the Euclidean distance. Usually, the training sample points are used as the center, therefore it can be stated: $c = x_s \;\wedge\; N_c = N_s$.

$$\tilde{y}(x_0, x_s, y_s) = \Psi_0^T w = \sum_{i=1}^{N_c} w_i \,\psi(\,||x_0 - c_i||\,) \tag{2.36}$$

The vector of the unknown coefficients w, is obtained by solving the system of linear equations given in equation (2.37). The gram matrix Ψ is defined as $\Psi_{ij} = \psi(||\,x_{s_i} - x_{s_j}\,||)$. In other words, the gram matrix, is the kernel function evaluated at the Euclidean distance between the i^{th} and the j^{th} samples.

$$\Psi\, w = y_s \tag{2.37}$$

The *missioninformer* invokes Pythons library *SciPy* and *SMARTy* for the RBF calculations. *SciPy* enables the user to define 7 different kernels, which are given in the equations (2.38) to (2.44), by changing one input parameter [3]. *SMARTy* uses the Thin Plate Spline (TPS) kernel. The variables r, ϵ are denoted as the Euclidean distance and an adjustable constant for Gaussian or multiquadtratic functions, respectively. Only 6 of the 7 kernels were used, since the quintic kernel took multiple hours to provide a solution. The 6 kernels are used for a comparison to Kriging. The results of the comparison are shown in 3.4

$$multiquadric = \sqrt{\left(\frac{r}{\epsilon}\right)^2 + 1} \tag{2.38}$$

$$inverse = \frac{1}{\sqrt{\left(\frac{r}{\epsilon}\right)^2 + 1}} \tag{2.39}$$

$$gaussian = exp\left[-\left(\frac{r}{\epsilon}\right)^2\right] \tag{2.40}$$

$$linear = r \tag{2.41}$$

$$cubic = r^3 \tag{2.42}$$

$$quintic = r^5 \tag{2.43}$$

$$thin\ plate\ spline = r^2\, ln(r) \tag{2.44}$$

2.5.3 Investigations on different surrogate models

In this subsection, the procedure, which was undertaken to investigate on the two most widely used surrogate models RBF and Kriging shall be explained. Note, the results are all presented in section 3.4. In the beginning of this section an introduction to RBF and Kriging and some reasons for the necessity of a reliable surrogate model were given. However, the main reason comes through the high dependency of the *missioninformers* output, cruise fuel mass and their gradients. As explained in the same section, the number of the invocation of the surrogate models is very high.

Only with a surrogate model, which replicates the underlying physical based behavior correctly, the results of *missioninformer* can be considered correct as well. Due to this heavy dominance 8 different options for Gaussian Kriging and RBF with TPS were chosen as the investigation parameter, see table 2.2. For further reading, Gaussian Kriging will be referred to as normal Kriging or only Kriging and RBF with TPS as TPS. The 8 options for Kriging and TPS were equivalent and are given in table 2.2. The short names on left side of the table defines the option's shortcut and are important for interpreting the plot results in section 3.4. The values $-1, 0, 1, 2$ for Augmentation mean no, constant, linear, and quadratic trend function, respectively. In case Regularization is set to True, a regularization constant is added to each control point.

Name	**Augmentation**	**Regularization**
A	-1	False
B	0	False
C	$+1$	False
D	$+2$	False
E	-1	True
F	0	True
G	$+1$	True
H	$+2$	True

Table 2.2: *SMARTy* Kriging and TPS parameter used for investigation surrogate quality

The investigations were performed with two different missions as input and are reproduced in table 2.3. The masses are given in kilogram and the cruise altitude h and cruise range R_{cr} in meters. For the central difference step-size the default value of 1e−5 was chosen. First the surrogate models Kriging, TPS and RBF were only tested for the state condition, meaning no gradients were calculated.

Mass	**Mission 1**	**Mission 2**
maximal takeoff	245e3	245e3
maximal landing	192.2e3	192.2e3
operating empty	132.5e3	132.5e3
manufactures empty	119.2e3	119.2e3
maximum zero fuel	180.5e3	180.5e3
maximum fuel	107.6e3	107.6e3
design payload	38.52e3	33.60e3
maximum payload	48.00e3	48.00e3
Flying parameters		
Ma	0.83	0.82
h	10.668e3	11.000e3
R_{cr}	9186.0e3	5185.6e3
Others		
step-size	1e−5	1e−5
weight-factor	0.7	0.3

Table 2.3: Missions definitions for investigation on different surrogate models

For *SciPy*'s RBF 7 different kernel functions, which are given in the equations (2.38) - (2.44), were tested with a *SciPy*s *smooth* value of 0.1. In case of *smooth* value greater zero, regularization is employed. Without it having defined to be greater zero, the generation of a surrogate model failed. For seeing the used *SciPy* RBF's kernel functions and interpreting the *SciPy* RBF investigation results the table 2.4 is provided. Note, the kernel quintic was dropped due its high amount of calculation time.

Name	**Kernel**
A	multiquadric
B	inverse
C	gaussian
D	linear
E	cubic
F	thin plate

Table 2.4: Different *SciPy* RBF kernels used for surrogate model quality investigation

Since in the state calculation only 3 surrogate models ($\tilde{LoD}, \tilde{AoA}, \tilde{TSFC}$) are required to be generated, the research is conducted much faster compared to the gradient version. The weight before cruise or the starting mass for the cruise segment m_s, the fuel weight only for the cruise segment m_{cr}, the total fuel mass m_f, which is required for the whole mission and the number of the fix-point iterations are considered for the state investigation. For each mass the mean value and the standard deviation over the options given in the tables 2.2 and 2.4 following the equations (2.45) and (2.46), respectively were conducted.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{n} x_i \tag{2.45}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i}^{N}(x_i - \bar{x})^2} \tag{2.46}$$

The reason for monitoring the mentioned masses is as follows. Since only the cruise flight segment involves a physical based equation rather than the empirical collected fuel fractions, which are constants and thus their multiplication results to a new constant (commutative behavior), only a scaling difference between the masses is expected. The fuel fractions are used only before and after the cruise flight segment. Therefore, if mistakes occurs within the investigation, plotting all the masses enables to find them easily visually. Thus considering the mentioned masses can be viewed as a verification technique. The second reason is, the impact of the different surrogate model options shall not only be looked at the final outcome, total fuel mass f_m, but also on the masses in between.

The number of fix-point iterations for the state allows making assumptions about the execution time. In general, it can be assumed, the less the number of fix-point iterations, the less the execution time is. Thus, it provides information about the impact of the model parameter on the execution time and convergence behavior. For only Kriging, 8 options were tested, therefore 8 results are obtained. From here it is possible to compare only within the Kriging options. For TPS the same is done with its own 8 results. For RBF 7 different kernel versions were tested. With the presented method the surrogate model can only be compared within its own surrogate model options environment. However, it is desired to compare Kriging with TPS and with *SciPy*s RBF, which would result in $8+8+7 = 23$ results to be compared. Note, for further reading RBF will be used as an abbreviation for *SciPy*s RBF. Since this is too much for one single plot, the mean value for each surrogate model (Kriging, TPS, RBF) is calculated and compared. In case the mean values matches or exhibits a low deviation it can be assumed that the corresponding models derive to same or resembling solutions and vice versa.

By looking at the standard deviation within each model, a robustness analysis can be performed. In case the standard deviation (std) is low, the respective surrogate model is not influenced much by the input parameters and thus can be seen as stable. A stable or robust might not be the most accurate model, however, it can be assumed to be more reliable for unknown complex underlying functions. Also, the robustness of the surrogate models (Kriging, TPS, RBF) was compared among each other. All investigations were performed for both missions separately and together. For the gradient version, the same is done plus the execution time for the whole process including reading the aerodynamic input files and writing out all the gradients.

The next step was to explore the accuracy of the surrogate models. For this purpose, a set of 38 training data was provided. With this set 3 different accuracy investigations were employed. In the first version 20 sample points were removed and the surrogate models were trained thus with the remaining 18 sample points (V18). The second version, 10 sample points were removed and the surrogate models were trained with 28 sample points (V28). In these both cases, the obtained model was invoked at the removed samples locations. The actual functions value for $\tilde{LoD}$, $\tilde{AoA}$, $\tilde{TSFC}$ at the removed values are known, since they were part of the initial 38 sample points. In order to evaluate the error for the surrogate models, RMSE according to equation (2.29) was used. For the third investigation, the *Leave-one-out cross-validation* (LOOCV) was performed. The idea of this method is, one sample point is

left out and with the remaining sample points a surrogate model is trained. In case of 38 sample points (V38) there are 38 ways to leave one sample point out and thus 38 different surrogate models can be generated. Each generated surrogate model is invoked at the location of the left out sample point and by exploiting RMSE the error was measured. To be more precise, *missioninformer* automatically finds non trimmed sample data and does not include it for generating surrogate models. For this workflow, instead of the overall 38 sample points, only 35 fully trimmed sample points, were used.

For the 3 mentioned versions, each version can be tested only with regard to the surrogate model (Kriging, TPS, RBF) and inside a respective surrogate model there are 8 options for Kriging and TPS and 7 options for RBF. However, because bad execution time was observed when using *SciPy*s RBF, only Kriging and TPS were chosen for the mentioned investigation. In case of RBF's quinitc kernel, even after 4 hours no solution was obtained. The explained process is performed for the state and the gradient version. The options which exhibit the least RMSE are collected and presented in the results subsection in 3.4.1.

2.6 Shape Gradients

In this section it shall be explained, how the gradients of the presented cruise fuel burn equation (ODE in equation (2.11)), can be computed. When the gradients are mentioned, then gradients with respect to the shape parameter, which are used to define the aircraft wing, are meant. For this work two main methods were evolved, an analytical and a numerical approach. The first one, however showed not be successful. A possible explanation for that will be given in the upcoming section. The numerical approach is based on the *Central differencing Scheme* and is divided into 2 different workflows called direct and indirect, which will be elaborated in subsection (2.6.2). Furthermore, a step-size study in order to find a reasonable step-size and thus reliable gradients will be topic of the subsection 2.6.2

2.6.1 Analytical attempt

Assuming, an analytical solution is possible, a fast and maximal computational accurate solution is obtained. These were the main reasons for trying to find an analytical solution. Fast in this regard means, the result is received immediately after all mathematical operations (summation, multiplication) are executed without any necessity of additional repetitive loops. The state equation for the cruise fuel burnt is written below again and for simplicity will be referred to as ODE and the objective is to find $dODE$.

$$ODE = \frac{dm_{cr}}{ds} = \frac{g}{a}\,\frac{1}{Ma}\,TSFC\,(m_s - m_{fe})\,\frac{LoD\,tan(\theta) + 1}{LoD\,cos(AoA) + sin(AoA)}$$

The ODE's gradient $dODE$ with respect to the shape parameters is already given in [14]. However, the same work has been done again in order to verify the existing solution. Once by hand and once by using Python's library *SymPy* for a symbolic solution. Ilic [14] results could be proven to be entirely correct and shall be stated in the upcoming equations. Since the gradient version is a term loaded equation, it will be split into groups, as advised in [14]. The grouped equation as the new basis is given in equation (2.47) for which its terms definitions are provided in equations (2.48) to (2.50)

$$ODE = k_c\,k_a\,k_m \tag{2.47}$$

$$k_c = \frac{g}{a}\frac{1}{Ma}\,TSFC \tag{2.48}$$

$$k_a = \frac{LoD\,tan(\theta)+1}{LoD\,cos(AoA)+sin(AoA)} \tag{2.49}$$

$$k_m = m_s - m_{cr} \tag{2.50}$$

A change in the shape parameter p results into a change in $TSFC$, m_s, m_{cr}, LoD, AoA, which means these variables are not to be considered as constants. Equations (2.51) to (2.56) shows the general form of the $dODE$, where the flight path angle $\theta \neq 0$ is not necessarily zero. The equations (2.53) and (2.54) results from the steady state flight condition. Here the denoting of the variables q, S, W , L , D , T is the dynamic pressure, reference wing area, weight, lift, drag and thrust, respectively.

$$\frac{dk_a}{dp} = \frac{[tan(\theta)\,sin(AoA) - cos(AoA)]\,\frac{dLoD}{dp} + [LoD\,sin(AoA) - cos(AoA)]\,[LoD\,tan(\theta)+1]\,\frac{dAoA}{dp}}{[LoD\,cos(AoA)+sin(AoA)]^2} \tag{2.51}$$

$$\frac{dLod}{dp} = \frac{d}{dp}\frac{C_L}{C_D} = \frac{1}{C_D^2}\left(C_D\frac{dC_L}{dp} - C_L\frac{dC_D}{dp}\right) \tag{2.52}$$

$$L = W\,cos(\theta) - [D + W\,sin(\theta)]\,tan(AoA) \tag{2.53}$$

$$C_L = \frac{m\,g}{q\,S}\,[cos(\theta) - sin(\theta)tan(AoA)] - C_D\,tan(AoA) \tag{2.54}$$

$$\begin{aligned}\frac{dC_L}{dp} = {} & \frac{g}{q\,S}\,[cos(\theta) - sin(\theta)\,tan(AoA)]\frac{dm}{dp} - \frac{m\,g}{q\;S^2}\,[cos(\theta) - sin(\theta)tan(AoA)]\frac{dS}{dp} \\ & - \frac{1}{cos(AoA)^2}\left(\frac{m\,g}{q\,S}sin(\theta) + C_D\right)\frac{dAoA}{dp} - tan(AoA)\frac{dC_D}{dp}\end{aligned} \tag{2.55}$$

$$\left(\frac{dC_L}{dp}\right)_{T=0} = \frac{g}{q\,S}cos(\theta)\frac{dm}{dp} - \frac{m\,g}{q\,S^2}cos(\theta)\frac{dS}{dp} \tag{2.56}$$

The *missioninformer* assumes the flight path angle θ to be zero and thus the above stated equations are simplified to the two upcoming equations. Even though it has been showed how LoD can be derived w.r.t. the shape parameter, however since its gradient is provided as input data, there is no need to its equation.

$$k_a(\theta = 0) = \frac{1}{LoD\,cos(AoA)+sin(AoA)} \tag{2.57}$$

$$\frac{dk_a(\theta = 0)}{dp} = \frac{-cos(AoA)\,\frac{dLoD}{dp} + [LoD\,sin(AoA) - cos(AoA)]\,\frac{dAoA}{dp}}{[LoD\,cos(AoA)+sin(AoA)]^2} \tag{2.58}$$

The mass term k_m was the obstacle, why the gradients can not be solved analytically as it will be shown now. The starting mass m_s can be modeled as done in equation (2.59), where m_{struc}, m_p, $m_{cr}(R)$, m_o are denoted as masses of structure, payload, cruise and other,

respectively. The structure and other masses as well the payload are not meant to change by changing the shape parameter. The only remaining mass, which is desired to change with the cruise range R is $m_{cr}(R)$. Transforming this idea to mathematical expressions, the equations (2.60) to (2.63) are obtained.

$$m_s = m_{struc} + m_p + m_{cr}(R) + m_o \tag{2.59}$$

$$\frac{dm_{struc}}{dp} = 0 \tag{2.60}$$

$$\frac{dm_p}{dp} = 0 \tag{2.61}$$

$$\frac{dm_{cr}(R)}{dp} \neq 0 \tag{2.62}$$

$$\frac{dm_o}{dp} = 0 \tag{2.63}$$

Following these the equation (2.64) is obtained and by inserting this to the mass definition from equation (2.50), the undesired state given in equation (2.65) can be observed.

$$\frac{dm_s}{dp} = \frac{dm_{cr}(R)}{dp} \tag{2.64}$$

$$\begin{aligned} \frac{dk_m}{dp} &= \frac{dm_s}{dp} - \frac{dm_{cr}(R)}{dp} \\ &\rightarrow \frac{dm_{cr}(R)}{dp} - \frac{dm_{cr}(R)}{dp} \\ &\Rightarrow 0 \end{aligned} \tag{2.65}$$

In this case the term for which the gradient is desired to be calculated vanished. Therefore, the approach stated in equation (2.66) was pursued instead, which leads to equation (2.67). In words, the change of the starting mass is assumed to be constant and thus is not influenced by the change of any shape parameter. At this point, it shall clearly be highlighted that this assumption is only made in order not to lose the cruise fuel mass term. Whether this models the physical behavior correctly was not known at this stage.

$$\frac{dm_s}{dp} = 0 \tag{2.66}$$

$$\frac{dk_m}{dp} = \frac{-dm_{cr}}{dp} \tag{2.67}$$

However, after having performed, the complete process for obtaining the analytical solution for the cruise fuel mass gradients, it can be stated that the assumption made in equation (2.66) is not correct. For this knowledge to gain, the gradients were calculated numerically using central finite differences. Due to the big deviation between the analytical and the numerical solution, the proposed finding can be given. Nevertheless, the further progress for receiving the final analytical solution shall be explained. The variables $m_{f,zero}$, f_{reser}, m_f for the upcoming equations are denoted as zero fuel mass, fuel reserve mass, and total fuel for all flight segments, respectively. As explained in section 2.4 the variable starting with fr_i are

fuel fractions, where fr_1 , fr_2 , fr_3 , fr_4, fr_5 , fr_6 , fr_7, fr_{cr} are for the flight segments, engine start, taxi to runway, takeoff, climb and acceleration, taxi to parking, landing, descent and cruise fuel fraction respectively.

$$fa\,(fr_{cr}) = f_{zero}\, fr_{cr}\, fr_1\, fr_2\, fr_3\, fr_4\, fr_5\, fr_6\, fr_7 \tag{2.68}$$

$$m_f = \frac{fa(fr_{cr}) - m_{f,zero}}{f_{reser} - fr_{cr}\, fr_1\, fr_2\, fr_3\, fr_4\, fr_5\, fr_6\, fr_7} \tag{2.69}$$

$$fr_{cr} = \frac{m_s - m_{cr}}{m_s} \tag{2.70}$$

$$m_s = (m_{f,zero} + m_f)\, fr_1\, fr_2\, fr_3\, fr_4 \tag{2.71}$$

The objective is to solve the equation (2.69). However, the problem is that the three equations (2.69) to (2.71) are coupled. This can be seen better by rearranging the following equation.

$$ff = fr_1\, fr_2\, fr_3\, fr_4,\ fr_5\, fr_6\, fr_7 \tag{2.72}$$

$$\begin{aligned} m_f &= \frac{fr_{cr} fr_1\, fr_2\, fr_3\, fr_4,\ fr_5\, fr_6\, fr_7 - 1}{f_{reser} - fr_{cr},\ fr_1\, fr_2\, fr_3\, fr_4,\ fr_5\, fr_6\, fr_7}\, m_{f,zero} \\ &= \frac{fr_{cr},\ ff - 1}{f_{reser} - fr_{cr},\ ff}\, m_{f,zero} \end{aligned} \tag{2.73}$$

$$fr_{cr} = \frac{m_s - m_{cr}}{m_s}$$

$$m_s = (m_{f,zero} + m_f)\, fr_1\, fr_2\, fr_3\, fr_4$$

It can be observed that in equation (2.73), the output of the ODE, the fuel cruise mass m_{cr} is required. This, according to the equation (2.70), requires for its solution the cruise starting mass m_s. However, for getting the starting mass m_s, according to equation (2.71), the total fuel mass m_f is demanded. The problem can be understood visually quite easily and thus is provided in figure 2.4.

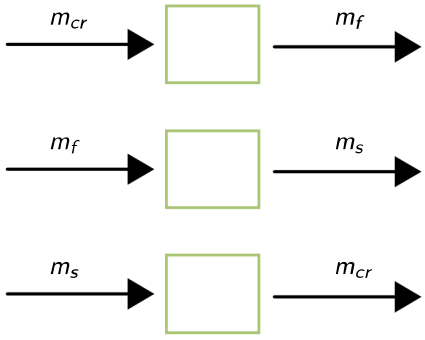

Figure 2.4: Coupled mass equations

One way to solve equation is to solve the equation (2.75) with another fix-point iteration. The used scalars are given in equations (2.72) and (2.74). The one disadvantage lies in the

nature of fix-point iterations, its repetitiveness, which requires higher execution time. The other is, that a more precise solution could be obtained.

$$ff_2 = fr_1\, fr_2\, fr_3\, fr_4 \tag{2.74}$$

$$m_f = m_{f,zero} = \frac{\left(1 - \dfrac{m_{cr}}{(m_{f,zero} + m_f)\; ff_2}\right) ff - 1}{f_{reser} - ff\left(1 - \dfrac{m_{cr}}{(m_{f,zero} + m_f)\; ff_2}\right)} \tag{2.75}$$

However, with some trial and error this coupling could be resolved and a without the necessity of fix-point iterations could be found, which is given in equation (2.76). The form of the solution, which was found by hand, originally contained a square root. By making use of Matlab and Pythons library *SymPy*, a symbolic solution could also be obtained. Matlabs and *Sympy*s solution were equivalent in their form, thus also in their results, when inserting values for the variables. However, their form of solution did not contain any square root. The stated equation (2.76) was used for further progress, since it is assumed to be easier for deriving gradients.

$$m_f = \frac{ff_2\, m_{f,zero} + ff\, m_{cr} - ff\, ff_2\, m_{f,zero}}{ff_2\,(ff - f_{reser})} \tag{2.76}$$

In order to highlight why equation (2.76) is easier for deriving gradients, the considered constants are placed on the left side and the depended or relevant part for derivation are on the right side in equation (2.77)

$$m_f = \frac{ff_2\, m_{f,zero} - ff\, ff_2\, m_{f,zero}}{ff_2\,(ff - f_{reser})} + \frac{ff\, m_{cr}(p)}{ff_2\,(ff - f_{reser})} \tag{2.77}$$

Recall, m_{cr} is the fuel mass for the cruise segment and is the output of the ODE, which now is desired to be derived with respect to the shape parameters. In equation (2.77), the term $m_{cr}(p)$ is indicated to be the only important parameter for the derivation. Therefore, the derivative of the total fuel mass can be written as in equation (2.79), where the mathematical definition for $dODE$ is given in equation (2.78).

$$dODE = \frac{d}{dp}\, ODE = \frac{d}{dp}\left(\frac{dm_{cr}}{ds}\right) \tag{2.78}$$

$$\frac{dm_f}{dp} = \frac{ff}{ff_2\,(ff - f_{reser})} \cdot dODE \tag{2.79}$$

Finally, by inserting all equations into each other the $dODE$'s final version is given equation (2.80), where $LoD, AoA, TSFC$ and their gradients are obtained by invoking the surrogate models, k_a , k_m, k_c are found in equations (2.57), (2.50) and (2.48), respectively. The gradients $\frac{dk_a}{dp}$, $\frac{dk_m}{dp}$ are found in equations (2.58) and (2.67), respectively. The variable p here is defined as a scalar for the sake of simplicity. In application p in the equations (2.78) to (2.80) is a vector. Thus, the equations need to be solved for each component of p.

$$
\begin{aligned}
dODE = \frac{ODE}{dp} &= \frac{d}{dp}\left(\frac{dm_{cr}}{ds}\right) \\
&= k_a\, k_m\, k_c\, \frac{dTSFC}{dp} + k_c\, TSFC\, k_a\, \frac{dk_m}{dp} + k_c\, TSFC\, k_m\, \frac{dk_a}{dp} \\
&= k_c \left(k_a\, k_m\, \frac{dTSFC}{dp} + TSFC\, k_a\, \frac{dk_m}{dp} + TSFC\, k_m\, \frac{dk_a}{dp} \right) \\
&= k_c \left(k_a\, k_m\, \frac{dTSFC}{dp} + TSFC\, k_m\, \frac{dk_a}{dp} \right) - k_c\, TSFC\, k_a\, \frac{dODE}{dp} \\
&= k_c\, \frac{\left(k_a\, k_m\, \frac{dTSFC}{dp} + TSFC\, k_m\, \frac{dk_a}{dp} \right)}{1 + k_c\, TSFC\, k_a}
\end{aligned}
\tag{2.80}
$$

After having seen the formal equations, there are two possible ways to calculate the gradients, the one-shot method and the indirect method. Both shall be explained with their respective workflow in figures 2.5 and 2.6. The one-shot version, depicted in figure 2.5, will let the state fix point-iteration finish as discussed in section 2.4. Based on the converged results it will then directly calculate the gradients for which only one evaluation is required. Since the state fix-point iteration is indispensable with effectively only one more function call the gradients are found.

The iterative workflow is depicted in figure 2.6. The initial starting and total fuel mass $m_{s,0}, m_{f,0}$ are guessed as explained in section 2.4. In order to calculate the gradient of the cruise fuel weight the initial cruise fuel weight and cruise starting mass is passed to the dODE-block, which calculates the gradient of the cruse fuel weight with respect to the shape parameters. This block is not required in the one-shot-version nor in the state fix-point iteration explained in section 2.4. Thus, here an additional calculation block is injected. It is used for computing the gradient of the total fuel mass. Now the whole process is repeated as long as the deviation between the previous and the current gradient of the total cruise fuel mass is below a defined threshold. Both methods were implemented in *missioninformer*, however they are not active, because of the explained reasons.

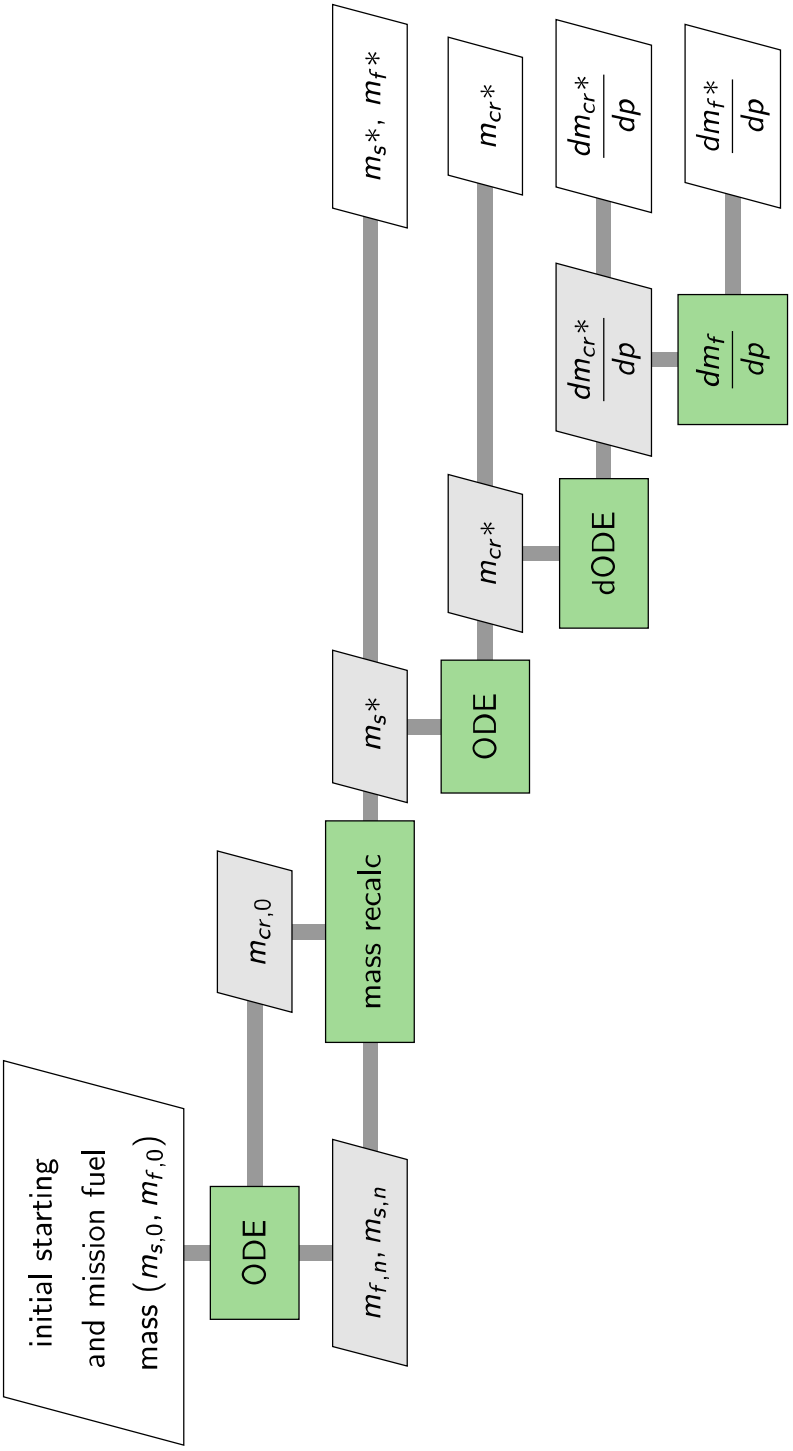

Figure 2.5: Analytical attempt to calculate gradients - one-shot version

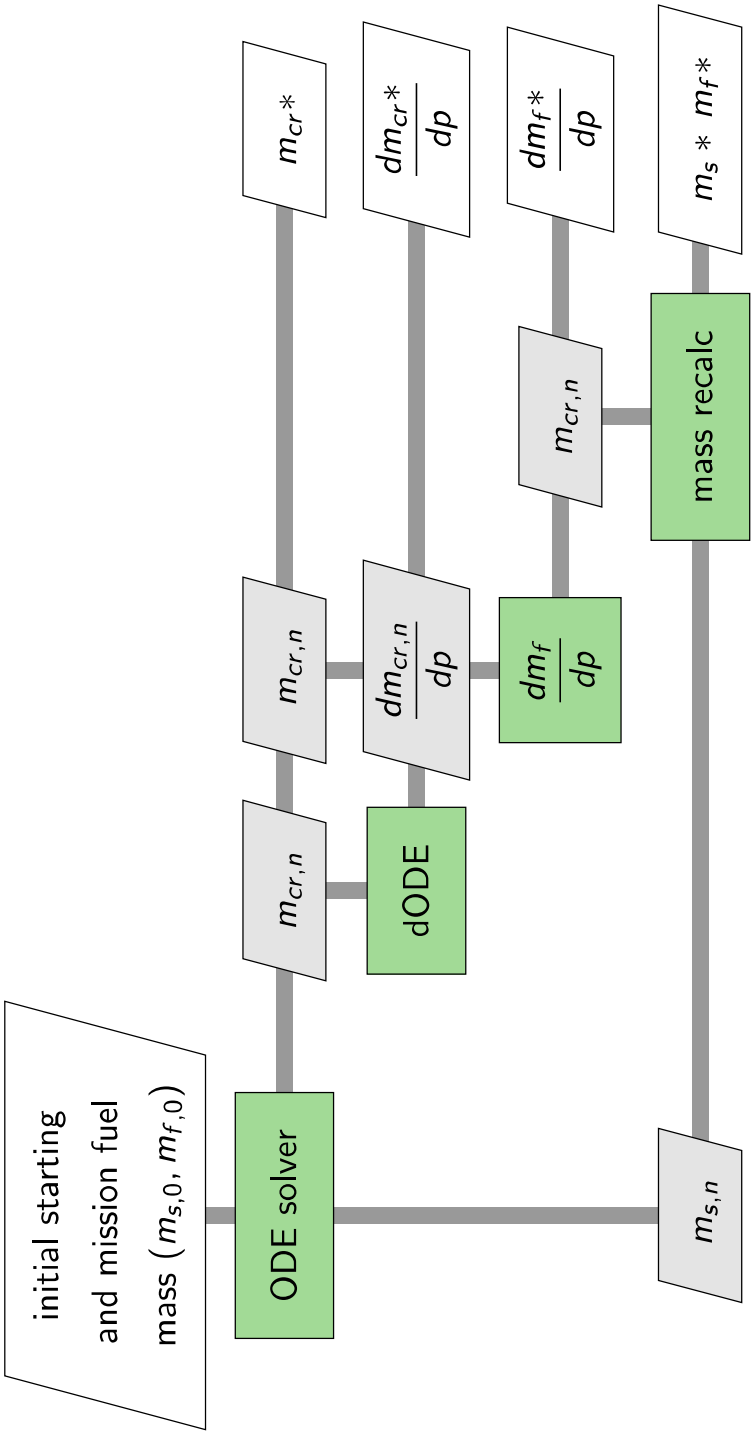

Figure 2.6: Analytical attempt to calculate gradients - iterative version

2.6.2 Numerical approach

As mentioned in the previous section, an analytical solution was pursued. However, the approach in section 2.6 had to be verified. In this subsection, the numerical approach for computing the gradients of the cruise fuel weight and the total fuel weight with respect to the shape parameters shall be explained. The three most frequently employed methods for deriving the gradients numerically are forward, backwards and central finite differences. In the majority of cases central finite differences or also known as central differencing scheme (CDS), which exhibits the best gradient computing accuracy. The equations for forward, backwards and central differencing are given in equations (2.81), (2.82) and (2.83), respectively.

$$\frac{df(x)}{dx} \approx \frac{\Delta f(x)}{\Delta x} = \frac{f(x + h_s) - f(x)}{h_s} \tag{2.81}$$

$$\frac{df(x)}{dx} \approx \frac{\Delta f(x)}{\Delta x} = \frac{f(x) - f(x - h_s)}{h_s} \tag{2.82}$$

$$\frac{df(x)}{dx} \approx \frac{\Delta f(x)}{\Delta x} = \frac{f(x + h_s) - f(x - h_s)}{2\,h_s} \tag{2.83}$$

Here the $f(x)$, h_s are denoted as any function with the input argument x and the step-size, respectively. The choice of the step-size h_s is important and thus will be dealt with later in more details. As pointed with the approximation sign ($\approx$), it is only an approximation of the gradient. The reason for this can be explained by viewing their derivation. In order to get the one of the three forms a given function is approximated by the Taylor series and since only an infinite long Taylor series can give the accurate solution, aborting the Taylor series after one or two terms cannot be accurate. Here it can also be observed that the gradients only around the approximated location can be considered acceptable. In other words, the higher the step-size h_s gets, the lower the quality of the approximation gets. The approximation error is also called truncation error. Since computers are used for calculating the derivate another error must be considered. The round off error, occurs due to the nature of storing digits into the computer memory or RAM. Having a decimal number, it consists of multiple digits. Each digit claims some memory and in case providing all numbers consisting of many digits all desired memory, two obstacles are faced. The memory capacities can be overwhelmed quite easy. The second problem is that, if only one number already contains more than 100 digits, it's processing by the hardware runs into infeasibly high execution time. Therefore, state-of-the-art accuracy uses double precision to store numbers. Already here an approximation of the real number is done. This computer based error is called round off error.

Up to now, two kinds of errors were introduced. The third error is called the abortion error. An infinite Taylor series would remove the abortion error, thus it would be vanished. In the introduction the forward and backwards finite differing methods were stated to be less accurate than the central finite differencing scheme. The reason for that can be seen by their derivation. In short, forward and backwards differing are methods of first order, whereas the central differencing scheme is a method of second order accuracy.

Since the CDS is considered to be more accurate in most applications, it is implemented in *missioninformer*. The equation (2.83) shows that CDS can be used as a block-box method, where f can by any computation in *missioninformer* and $x + h_s$, $x - h_s$ represents the parameter, for which the gradient is desired. It needs to be changed by adding and subtracting the

step-size h_s. However, applying this theory is not straight forward for the calculations done in *missioninformer* and thus the CDS does not act as a black-box function. The reasons for that are the subject of the upcoming discussion.

The gradient of the total fuel mass m_f with respect to the shape parameter is the object of desire $\frac{dm_f}{\vec{p}}$. The reason why the shape parameter variable $\vec{p}$ is chosen to be formulated as a vector is to highlight, that not only one gradient is calculated. Because no optimization can be accomplished with having defined only one shape parameter, it is more correct to talk about the shape parameter vector $\vec{p}$, which notation from here on will be continued to be followed. In order to apply the straight forward CDS as described with equation (2.83), the relationship for each variable LoD, AoA, $TSFC$ to all the shape parameters $\vec{p}$, as stated in equation (2.84) must be known.

$$LoD(\vec{p}), \quad AoA(\vec{p}), \quad TSFC(\vec{p}) \tag{2.84}$$

However, the encountered problem is that these relationships from equation (2.84) are not known, thus the black box straight forward CDS presented in equation (2.83) cannot be applied. The following simple and mathematical correct workaround was found.

$$LoD_{pertubated} = h_s \frac{dLoD}{d\vec{p}} \tag{2.85}$$

$$AoA_{pertubated} = h_s \frac{dAoA}{d\vec{p}} \tag{2.86}$$

$$TSFC_{pertubated} = h_s \frac{dTSFC}{d\vec{p}} \tag{2.87}$$

With these background, two different workflows, for solving the gradients of the total fuel mass with respect to the shape parameters can be discussed. The difference in both occurs in using the auxiliary cruise fuel fraction term fr_{cr} given in the following equation or inserting it directly into the remaining equations as presented in section (2.4)

$$fr_{cr} = \frac{m_s - f_{cr}}{m_s} \tag{2.23}$$

In case of direct insertion, the equation (2.76) is obtained, as explained in section 2.6 and is referred to as *direct method*

$$m_f = \frac{ff_2\, m_{f,zero} + ff\, m_{cr} - ff\, ff_2\, m_{f,zero}}{ff_2\,(ff - f_{reser})} \tag{2.76}$$

For defining the fuel cruise fraction fr_{cr} explicitly the method is referred to as *indirect method*. The indirect method's solution is given in equation (2.88), which derivation can be seen in section 2.4 with the equations (2.20) to (2.24).

$$m_f = \frac{fa - f_{zero}}{f_{reser} - fr_{cr}\, fr_1\, fr_2\, fr_3\, fr_4\, fr_5\, fr_6\, fr_7} \tag{2.88}$$

After analyzing two main differencing could be observed. The direct method often claims more fix-point iterations. In contrast, it was found to be more accurate by satisfying lower convergence criteria. In case of low convergence parameter, the indirect method stucked inside a loop, where it jumped between two values. Since the *missioninformer* has a limit of maximal allowed iterations of 500, the iterative method would stop after 500 iterations and

thus would require more iterations than the direct method. However, a relative and absolute tolerance [2] value of 5e−5 is assumed to be accurate enough. With this tolerance value, no permanent stay for the indirect method inside a loop could be observed. Also, in cases where the missions were defined reasonably, only 4 iterations were required for obtaining the gradient of the total fuel mass $\frac{dm_f*}{d\bar{p}}$ per gradient component and perturbation direction. Therefore, the *missioninformer* default method is chosen to be the indirect method.

The parameter which defines the quality of the gradients by applying CDS is the choice of the step-size h_s. In order to find an appropriate h_s two different missions were defined. One, which raised the *constraint violation check* to output warning statements and one without. Thus, these missions can be considered to be hard to be solved (mission 1) and the opposite (mission 2), respectively. For both missions, applying both methods (direct and indirect) a step-size study is performed. In case of the harder to solve mission 1, the number of maximal fix-point iterations was reached. The missions definitions can be found in table 2.5. The difference between current table 2.5 and the table 2.3 from subsection 2.5.3 is only in the definition of the range for mission 1. In table 2.5 this has been reduced by 1000 meters.

Mass	**Mission 1**	**Mission 2**
maximal takeoff	245e3	245e3
maximal landing	192.2e3	192.2e3
operating empty	132.5e3	132.5e3
manufactures empty	119.2e3	119.2e3
maximum zero fuel	180.5e3	180.5e3
maximum fuel	107.6e3	107.6e3
design payload	38.52e3	33.60e3
maximum payload	48.00e3	48.00e3
Flying parameters		
Ma	0.83	0.82
h	10.668e3	11.000e3
R_{cr}	10186.0e3	5185.6e3
Others		
step-size	1e−5	1e−5
weight-factor	0.7	0.3

Table 2.5: Missions definitions for investigation on step-size

The convergence behavior of both missions and both methods is presented in section 3.2. However, the final result is that missionsinformers default step-size is chosen to be $h_s = 1e{-}5$.

3 Results

In this chapter the results of the methods described in previous chapter shall be presented. The first topic for this purpose is the solution of the state ODE from equation (2.11).

3.1 Solving the ODE

The main equation upon which's solution the remaining computations are based on is the ODE for the cruise fuel burnt mass m_{cr}.

$$\frac{dm_{cr}}{ds} = \frac{g}{a}\frac{1}{Ma}\,TSFC\,(m_s + m_{cr})\,\frac{1}{LoD\,cos(AoA) + sin(AoA)} \tag{2.11}$$

The used technique therefore is by calling SciPy's function *solve_ivp* [4]. The library comes along with different solvers and the solution of all were investigated. However only a selection of the most interesting results shall be presented here. Having compared all available methods inside *solve_ivp* it could be found out that *LSODA* and *RK45* offered the least difference to the analytical solution obtained by equation (2.19). Its derivation and explanations can be read in sections 2.3.

$$m_{fe}(s = R_{cr}) = m_s\left(1 - e^{\frac{-g}{a}\frac{1}{Ma}\,TSFC\,\frac{1}{LoD\,cos(AoA)+sin(AoA)}\,s}\right) \tag{2.19}$$

The analytical solution is only valid, when assuming that $LoD, AoA, TSFC$ are constant. Comparing the analytical solution with *SciPy*'s numerical solution by employing all available methods, it could be observed, that $LSODA$ exhibits best results. The figures from 3.1 to 3.4 depicts the analytical and the numerical solution on the left side and their deviation on the right side. One of the options within *LSODA* and *RK45* is to set points where an integration step shall be performed. The higher the number of the given calculation points is, the higher the accuracy of the outcome is. Figures 3.1 and 3.2 depict the results when the user exerted calculation points. From range $R = [0; 100]$ meter and for the last 100 meter of the cruise flight range, 500 uniformly calculation points were set. In between, $\frac{R_{max}}{8} = \frac{7297989}{8} = 912248$ calculation points were set. Even though a high number of evaluation points leads to a more accurate result, recall that for each additional evaluation point, the surrogate models needs to be invoked. Therefore, it is desired to keep the number of the control points low, if possible.

In figures 3.3 and 3.4, the same calculations are performed with *LSODA* and *RK45*, respectively, but with their methods default number of calculation points. The default number of calculation points is much lower, e.g. for *RK45* less than 20 sample points were required. Comparing all the 4 presented results it can be stated that the deviation or *delta* between the analytical and numerical solution is in the order of grams, which is sufficient for the accuracy requirement of the *missioninformer*. Even though *LSODA* delivers the best performance, since *RK45* requires less integrations steps points with barley less accuracy, *RK45* with its default method for setting calculation points is set as default in the *missioninformer*.

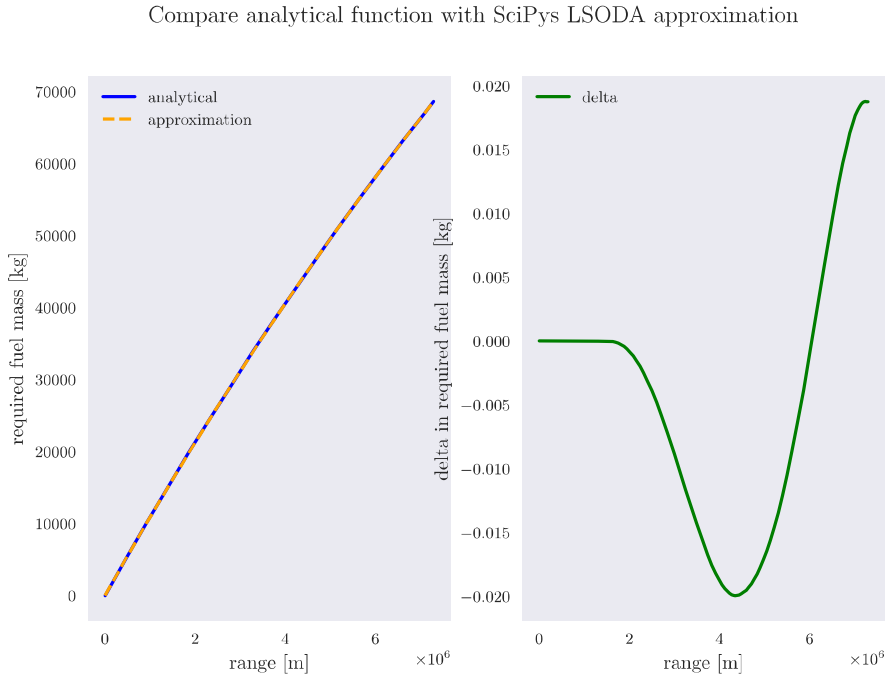

Figure 3.1: Solution obtained having used *LSODA* with user defined calculation points

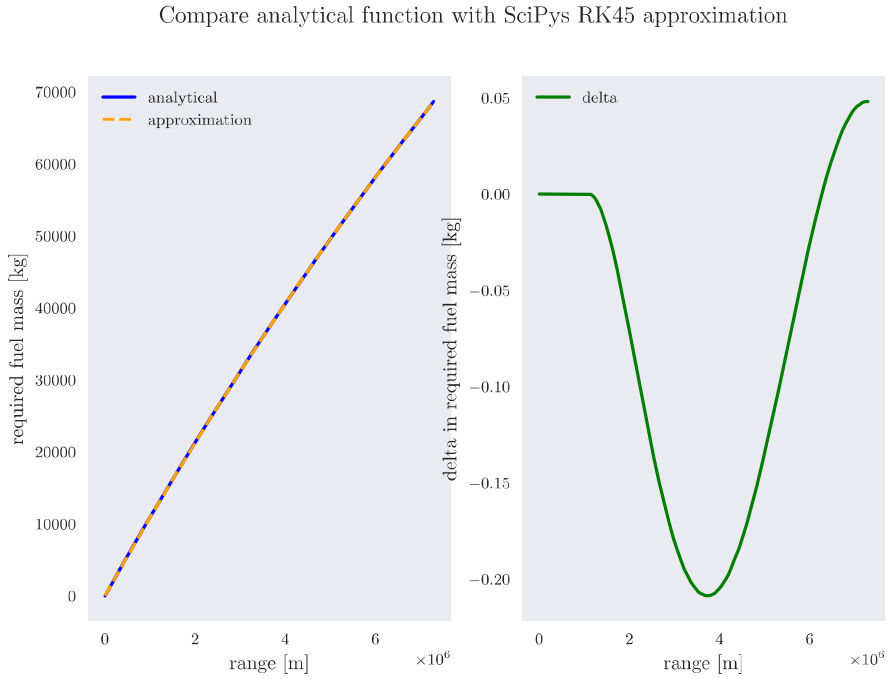

Figure 3.2: Solution obtained having used *RK45* with user defined calculation points

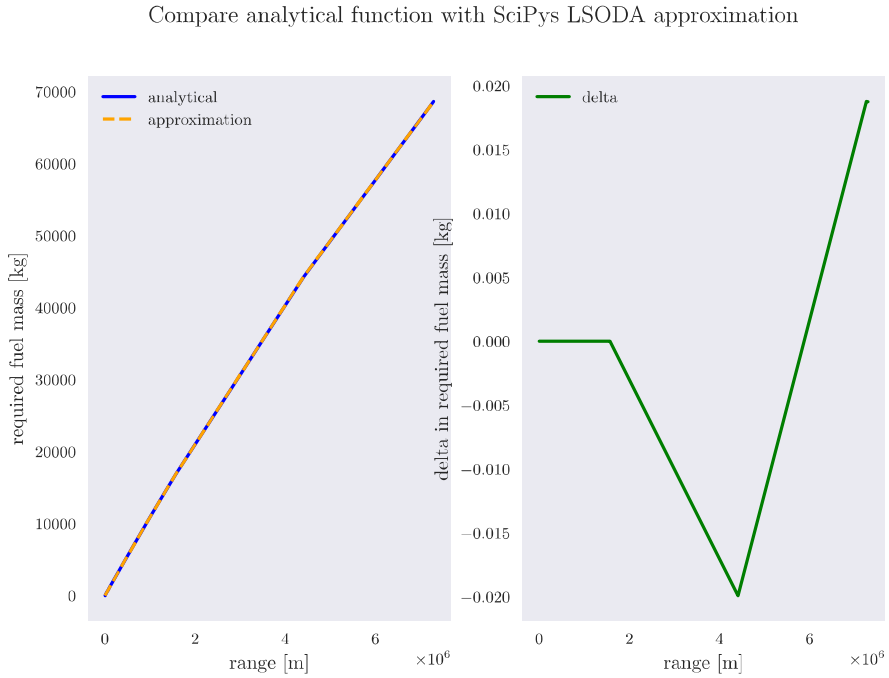

Figure 3.3: Solution obtained having used *LSODA* without user defined calculation points

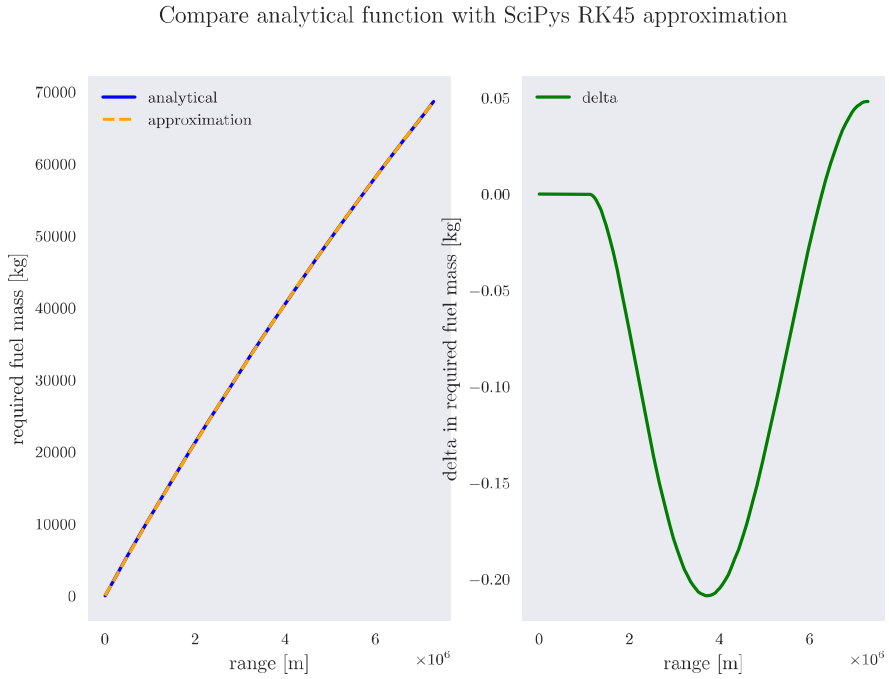

Figure 3.4: Solution obtained having used *RK45* without user defined calculation points

The results above were presented by assuming $LoD, AoA, TSFC$ to be constant. However, for *missioninformer* they are not constant, but dependent on Ma, h, m. In order to see the effect of these variables not to be constant, all available methods within *SciPy*s ODE solver were tested again. For testing purposes no real aerodynamic data output was used, but rather some reasonable values, which were only dependent on m, were generated. This data set was the input for a *1d* interpolation, which is also available in *SciPy*. Coming to the workflow, every time the ODE is solved, the values for the variables $LoD, AoA, TSFC$ are obtained by the mentioned 1d interpolation. This process can also be done with a lower or higher number of calculation points. However, no deviation between the lower and higher number of calculation points version, could be observed.

The figures 3.5 and 3.6 show the results, when the mentioned 1d interpolation and no interpolation is applied. On the left side, the default mechanism for choosing calculation points is plotted. On the right side, the above explained user defined calculation points were used for deriving the solution. Both sides show a deviation between the interpolation active and inactive versions, which is called *delta* in the mentioned figures. Figure (3.5) shows a strange and unexpected behavior in its delta. This is explained as follows. The number of calculation points for the active and inactive with the defaults' method to define calculation points is not equal. In the inactive interpolation version, the default option for defining calculation points requires less calculation points. However, for calculating delta, both number of calculation points mus be equal, to be valid. Therefore, the delta on the left side can not be considered to be correct.

This observation tells us, that *LSODA* is more flexible in increasing the number of calculation points than RK45. This can bee seen by viewing the delta on the left side of the figure 3.6. Here, a monotonically increasing delta can be identified. The default version of *RK45* took less than 20 calculation points, whereas *LSODA* required around 60 calculation points.

Interpolated $(LoD, AoA, TSFC)$ based on new current total mass vs non interpolated - LSODA

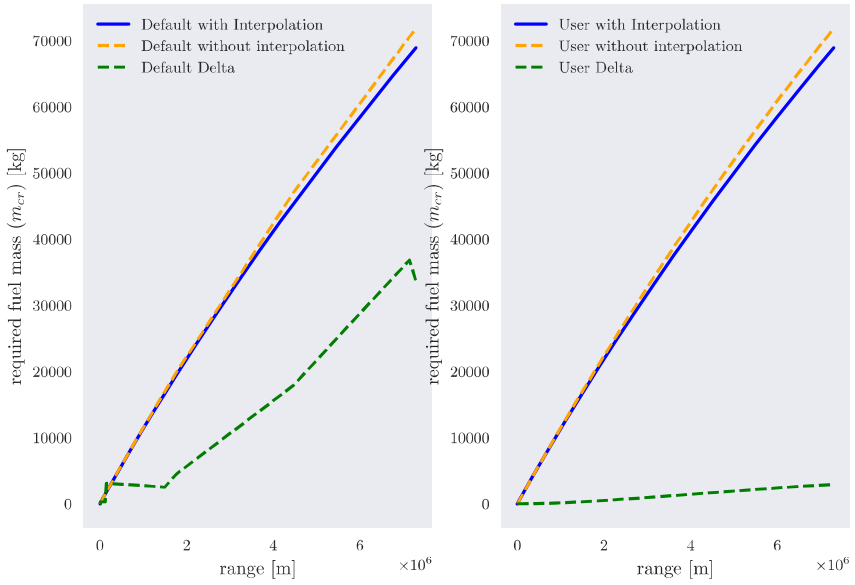

Figure 3.5: Solution obtained having used *LSODA* with and without user defined calculation points and active and inactive interpolation

Interpolated $(LoD, AoA, TSFC)$ based on new current total mass vs non interpolated - RK45

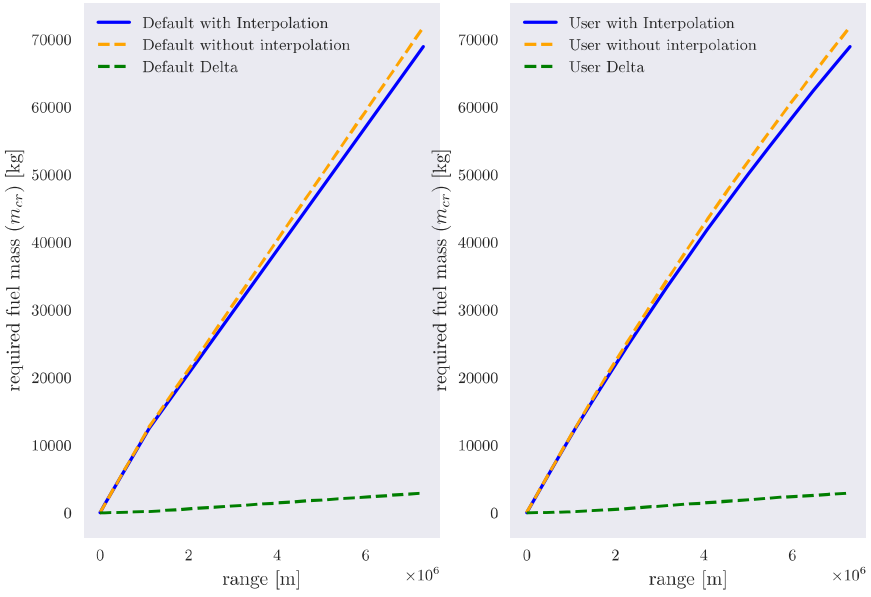

Figure 3.6: Solution obtained having used *RK45* with and without user defined calculation points and active and inactive interpolation

Comparing *LSODA* with *RK45* directly can be done via figure 3.7. Important for present concern is the right side, where the blue and orange graphs are the objects of our focus. However, the blue graph is not visible. The reason for that is, that is has been draw before the orange graph has been drawn. Therefore, the blue graph is beneath the covering orange graph. With this it can be said, no noticeable deviation in the results for active interpolation by having defined a ridiculously high number of calculation points can be seen. This leads to the following conclusion. No real difference in the accuracy of *LSODA* and *RK45* is exhibited, however, *LSODA* requires more calculation points. As a consequence of this, *RK45* can be chosen as the default method to solve the ODE in the *missioninformer*.

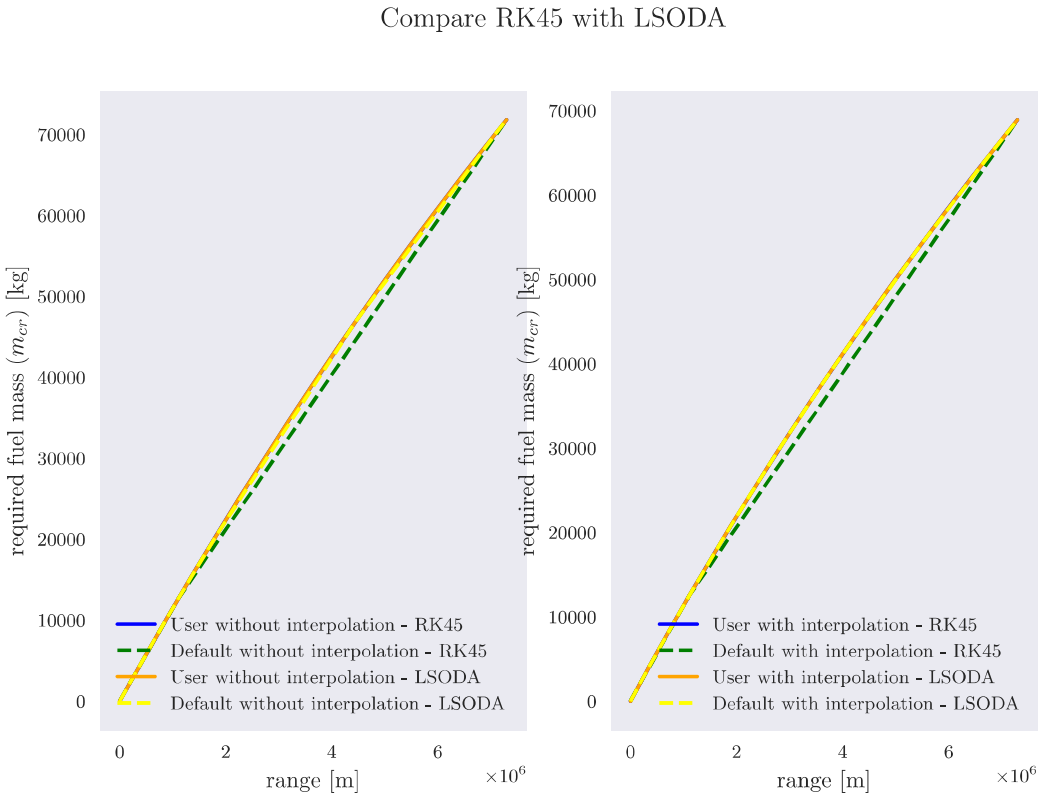

Figure 3.7: Direct compare *RK45* with *LSODA* for in/active and default/user versions

3.2 Conducting a step-size-study

In section 2.6.2 the reasons for the necessity of a step-size investigation has been described. In short it can be said, a wrong chosen step-size leads to not useable gradient values for the optimization. In this section the results of the performed step-size study shall be shown and elaborated. In order to calculate the gradients using different step-sizes, the procedure given in table 3.1 was employed. The left side shows the interval from which the step-sizes were taken, on the right side, the number of evenly distributed points, which shall be used as the step size, are given. The investigation was conducted for 2 different missions as explained in section 2.6.2. Both missions can be seen in table 2.5. Mission 1 encountered or failed the *constraint violation check*, which is explained in section 2.4. In cases, where the *constraint violation check* displayed warning messages, the respective mission required a high number of fix-point iteration. Mostly, the limit of maximal 500 allowed iterations was reached. In contrast, mission 2 successfully passed the *constraint violation check* and thus no warning messages were encountered. Also, the number of iterations were much lower.

Intervall	**# points**
[1e−1 ; 1e−3]	20
[1e−3 ; 1e−4]	10
[1e−4 ; 1e−5]	10
[1e−5 ; 1e−6]	15
[1e−6 ; 1e−7]	15
[1e−7 ; 1e−8]	10

Table 3.1: Chosen step-sizes

Furthermore, two different values for the tolerance parameter where used for the above-mentioned investigations. For the first examination the absolute and relative tolerance parameter [2] for the direct and indirect gradient calculation workflows, which are explained in section 2.6.2, are set equal to 5e−5. For the second study, for the direct method, the tolerance parameter were set to 5e−6, which can be done since the direct method can reach finer tolerances as explained in section 2.6.2. The CDS is computed in application for each shape parameter. In this study, only five selected parameters were investigated. All the upcoming results are achieved by employing *SciPy*'s RBF with multiquadric as kernel, which is given in equation (2.38), as the underlying surrogate model generator.

Note that not each analyzed result can be shown here, since the number of these would claim too many pages. The outcomes, which describes the gained finding will be highlighted instead. For this propose, the result will be reviewed for the direct and indirect method, where the direct method has a tolerance parameter of 5e−6. The results are only shown for mission 2 and for one shape parameter.

The first interesting finding is that both methods, direct and indirect, retain a so-called plateau with respect to step size variations. In the region of the plateau, the types of error, mentioned earlier, are in balance. In other words, within the plateau, no type of error dominates and the total value is small. In case of a too small step-size the round-off error becomes such high, that numerical instability is the consequence. In contrast, a too large step-size results in very high truncation error. In both cases, the result by using CDS cannot be considered to approximate the real gradient value. This plateau was searched for, in order

to be able to make statements and recommendations about the value for the step-size. This observation was seen in all tested cases for mission 2. The figure 3.8 and 3.9 show the plateau for the direct and indirect method for mission 2, respectively. It can be observed, that for both methods, direct and indirect, the plateau can bee seen around the step-size of 1e−5. This result was found also for remaining step-size studies the different shape parameters.

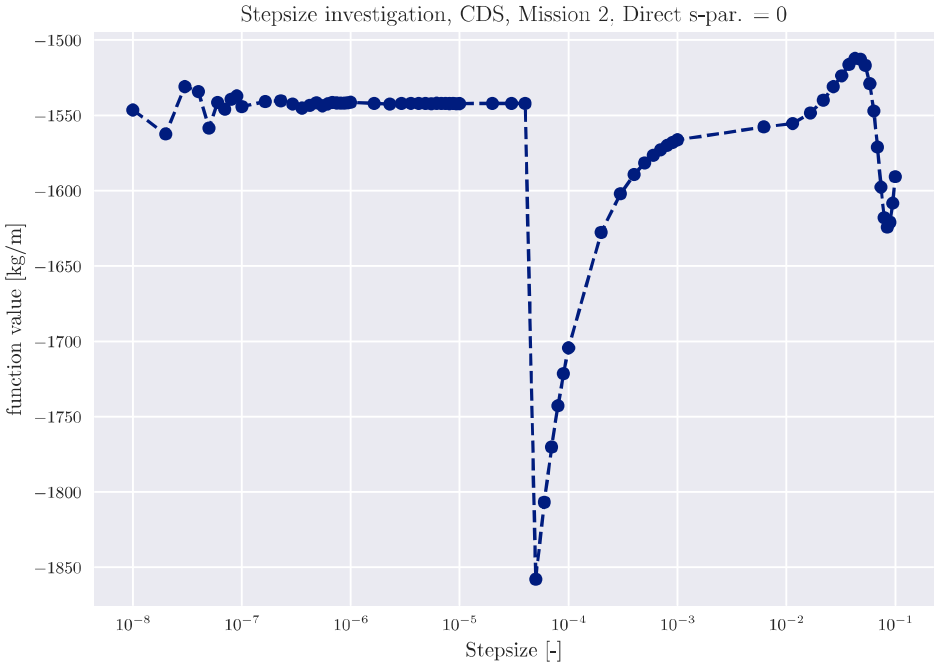

Figure 3.8: Step-Size-Investigation, Mission 2, Shape parameter 0, Applied method: Direct, Unequal tolerance parameter

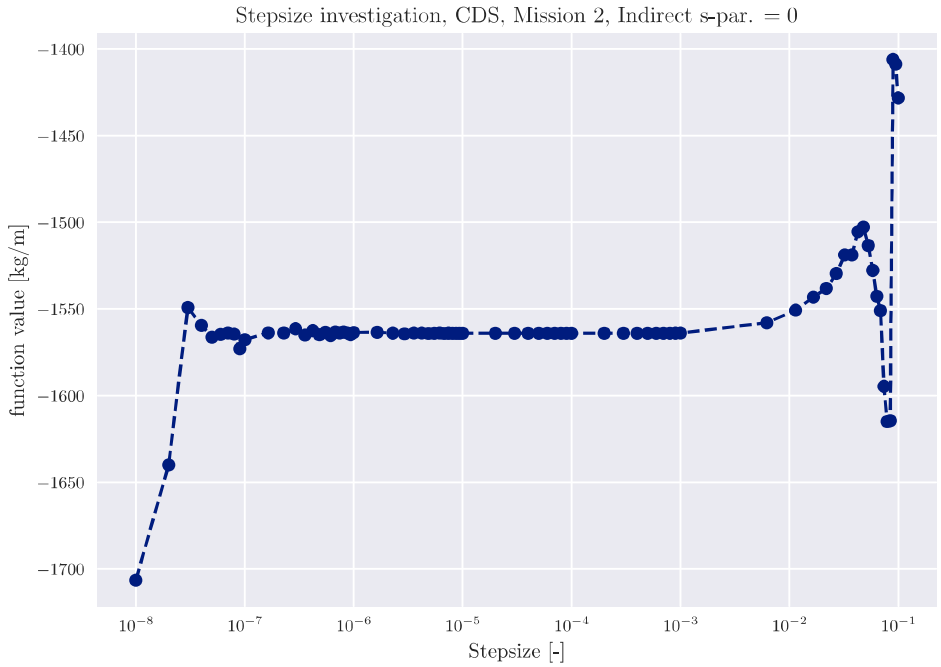

Figure 3.9: Step-Size-Investigation, Mission 2, Shape parameter 0, Applied method: Indirect, Unequal tolerance parameter

However, a clear plateau could only be detected for mission 2. The deviation of the gradient with changing step-size for mission 1 was in the orders of $10^7\ kg/m$ and $10^9\ kg/m$. This leads to the conclusion, when the *constraint check violation* is passed successfully, the step-size of 1e−5 can be used for the CDS gradient calculation. Otherwise, when receiving warning statements by the *constraint check violation*, not only can the step-size of 1e−5 not be used, rather CDS as the method for obtaining the gradients should not be used. This because, when no plateau can bee seen, there is no possibility to find a reasonable required step-size.

Another conclusion which can be made is by comparing the direct and indirect results. The indirect method seems to be more stable within a broader step-size interval. Also, this occurrence could be seen in for mission 2 for all tested shape parameters. Note the scaling of the y-axis, which defined as 3 times the standard deviation. With this in mind the stability of the gradient by changing the step-size within the plateau becomes more tangible.

As summary, it can be stated that the iterative indirect method was found to be more stable and choosing the step-size of 1e−5 could be proven to be optimal. This value neither leads to numerical instabilities nor does it not comply with the accuracy requirements. Therefore, the default method for solving the gradients with respect to the shape parameters is the indirect method with its default step-size of 1e−5.

3.3 Surrogate models

In this section different surrogate models, which are listed in section 2.5, will be explored with regard to their respective options. For generating the RBF surrogate model *SciPy* was used with its default kernel multiquadtratic, which is given in section 2.5.1 in equation (2.38). Therefore, a *smooth* value [3] of 0.1 was employed, which means regression is performed. Note for further reading, when the word RBF is mentioned, it always means *Scipy*'s RBF. Kriging surrogate models were generated by invoking *SMARTy*. Here the chosen default Kriging kernel was Gaussian, Augmentation and regularization were set to 1 and *True*, respectively. The training samples for generating the surrogate models were provided by the *DLR*'s analysis and optimization workflow *FSAerOpt* [25]. It is able to calculate fully trimmed states and consistent adjoint based gradients involving the flow solver *TAU*. A total of 38 training samples distributed in the Ma, h, m space using the Halton Design Of Experiments (DOE) method, which are re depicted in the figures 3.10 to 3.12, were provided for this and the upcoming surrogate models studies.

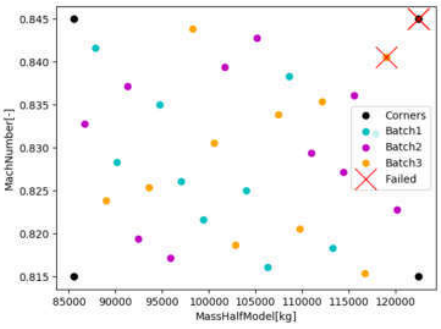

Figure 3.10: Halton DOE for half mass and Mach number ($m/2$, Ma)

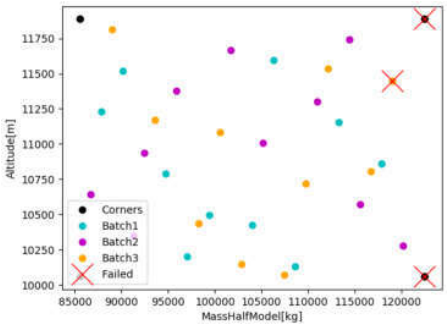

Figure 3.11: Halton DOE for half mass and altitude ($m/2$, h)

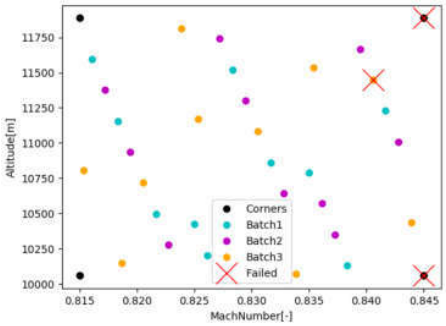

Figure 3.12: Halton DOE for Mach number and altitude (Ma, h)

The input for the surrogate model is 3 dimensional, meaning Mach numbers, altitudes and masses Ma, h, m are provided. With this set of input variables the desired predicted outcome for the output variables $LoD, AoA, TSFC$ are generated. Note that the output variables are called $LoD, AoA, TSFC$, but their respective interpolated values are denoted as $\tilde{LoD}, \tilde{AoA}, \tilde{TSFC}$. This means 3 different surrogate models are trained. Because of the 3 dimensional input and the 1 dimensional output, plotting the surrogate models would require 4 axes. Therefore, for showing results, plots at constant Mach numbers, altitudes and masses (Ma, h, m) are going to be presented. Also note that the mass, when solving the ODE is equivalent to the cruise starting mass m_s.

Since more research has been done than can be showed here explicitly with figures, only some results will be shown. This issue is understood better by highlighting the fact that the explorations are performed for state as well as the gradients. In case of the state surrogate models only 3 surrogate models are constructed. In case of the gradients, 126 shape parameter were given, thus $126 * 3 = 378$ surrogate models could be presented. Additionally, to recall, only sections where of Ma, h, m is constant can be visualized properly. Therefore, some results, which are supposed to summarize the findings, are chosen to be displayed. The structure of presenting the results is as follows. On the left and right side the RBF and Kriging surrogate interpolation models are depicted for the same constant parameter of of Ma, h, m, respectively. Therefore, interpolation models for $LoD, AoA, TSFC$ will be given at two different respective constant parameter values for Ma, h, m. In other words, for each input parameter Ma, h, m, one parameter is chosen to be constant at a specific value. This value is changed two times and isoplanes through the entire space of the interpolation models (RBF, Kriging) are sliced and shown next to each other. In total, 6 different constant values for each input parameter Ma, h, m were chosen. They are uniformly distributed between the lowest and highest values of the respective input parameter (Ma, h, m).

For interpreting the figures, be reminded that the angle of attack is also often referred to as $AOA = \alpha$ and note that the y-axis is equally color-coded. Figures 3.13 to 3.16 depict the interpolation models for the constant altitude 10058.4 m and 11887.2 m, for the output variables $LoD, AoA, TSFC$. In figures 3.13 and 3.14 noticeable differences in the interpolation models for AoA can be observed. The Kriging model on the right side of depicts a smoother transition than RBF. This observation can bee seen in all figures with a constant altitude from 3.13 to 3.24. This holds also, when varying the altitude or even the output variable. In other words, each interpolation model for the different desired output variables $LoD, AoA, TSFC$ confirms this observation. A possible reason for this occurrence could be that the Kriging value for regularization is automatically tuned in *SMARTy* and can be higher than the value for *SciPy*'s RBF. The answer to the question, which models predict the underlying physical phenomenons accurately requires a deep understanding of the relationship of all the input and output parameters and can be partly answered in subsection 3.4.1

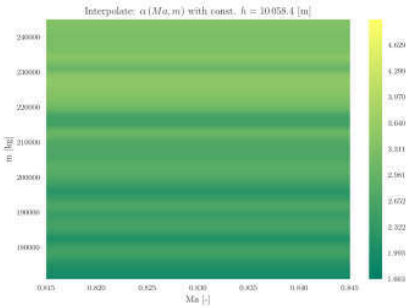

Figure 3.13: Surrogate interpolation model for AoA at constant altitude $h = 10058.4$ [m], RBF

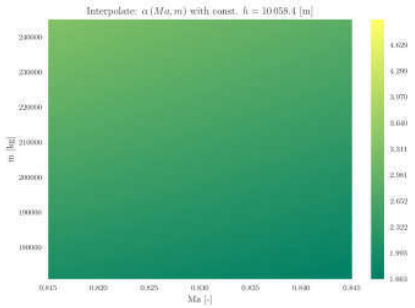

Figure 3.14: Surrogate interpolation model for AoA at constant altitude $h = 10058.4$ [m], Kriging

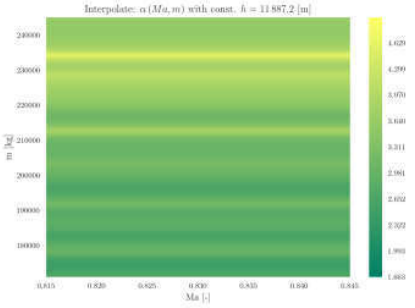

Figure 3.15: Surrogate interpolation model for AoA at constant altitude $h = 11887.2$ [m], RBF

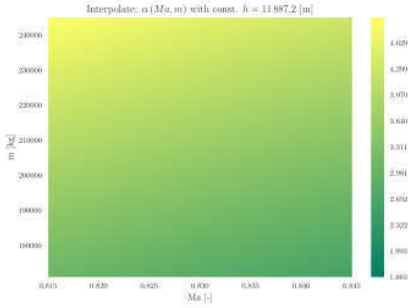

Figure 3.16: Surrogate interpolation model for AoA at constant altitude $h = 11887.2$ [m], Kriging

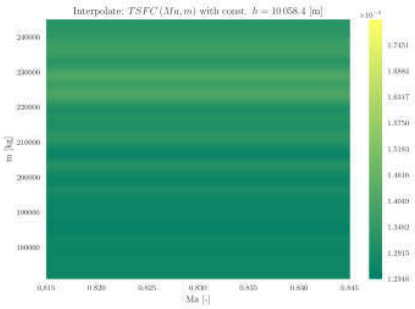

Figure 3.17: Surrogate interpolation model for $TSFC$ at constant altitude $h = 10058.4$ [m], RBF

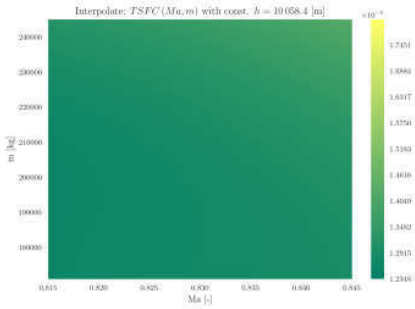

Figure 3.18: Surrogate interpolation model for $TSFC$ at constant altitude $h = 10058.4$ [m], Kriging

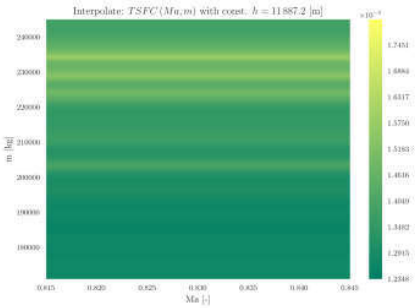

Figure 3.19: Surrogate interpolation model for $TSFC$ at constant altitude $h = 11887.2$ [m], RBF

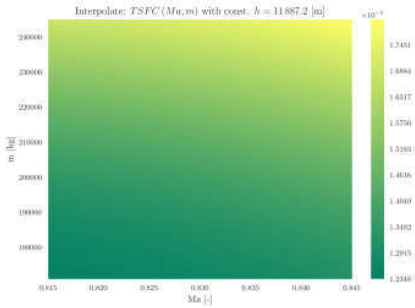

Figure 3.20: Surrogate interpolation model for $TSFC$ at constant altitude $h = 11887.2$ [m], Kriging

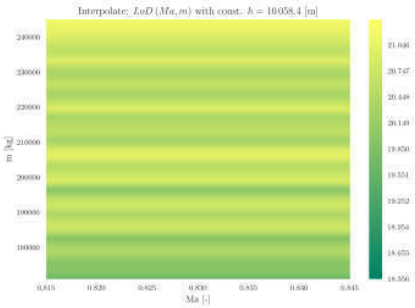

Figure 3.21: Surrogate interpolation model for LoD at constant altitude $h = 10058.4$ [m], RBF

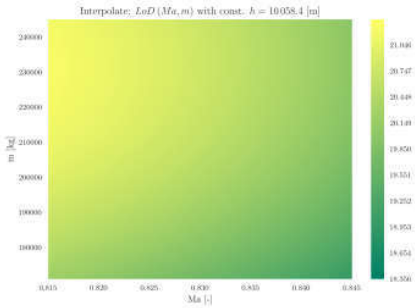

Figure 3.22: Surrogate interpolation model for LoD at constant altitude $h = 10058.4$ [m], Kriging

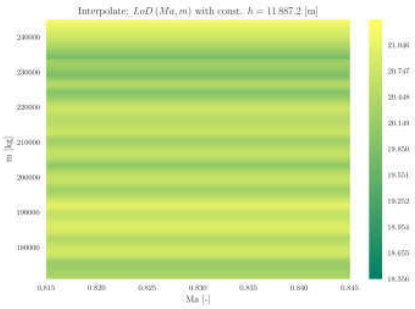

Figure 3.23: Surrogate interpolation model for LoD at constant altitude $h = 11887.2$ [m], RBF

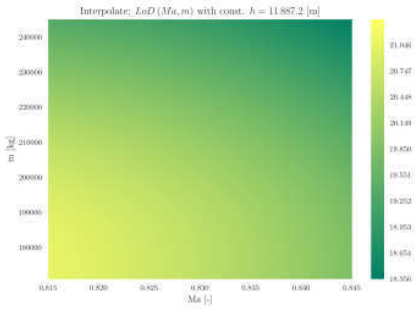

Figure 3.24: Surrogate interpolation model for LoD at constant altitude $h = 11887.24$ [m], Kriging

The figures from 3.25 to 3.32 show the interpolations models, when a cut at constant Mach numbers is performed. Here the same effect as described above for the constant altitude case can be observed. Kriging is smoother in the curve of its predicted values than RBF is. However, in the case of constant Mach numbers, one more phenomena can be observed. The actual values of for the desired output variables $LoD, AoA, TSFC$ are different when comparing RBF (left side) with Kriging (right side). This effect is clearly visible, e.g. in figures 3.29 and 3.30.

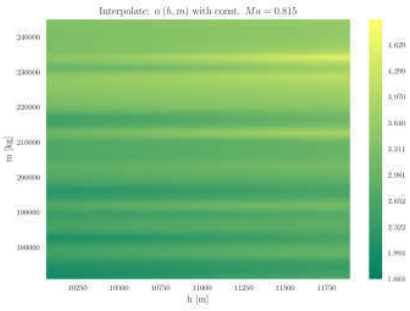

Figure 3.25: Surrogate interpolation model for AoA at constant $Ma = 0.815$ [-], RBF

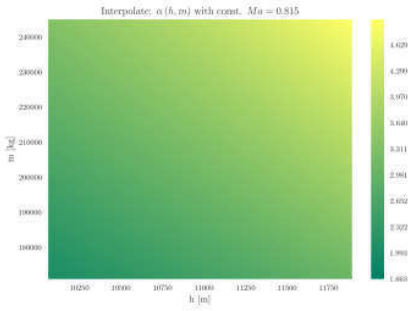

Figure 3.26: Surrogate interpolation model for AoA at constant $Ma = 0.815$ [-], Kriging

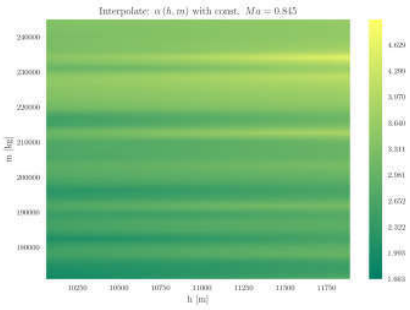

Figure 3.27: Surrogate interpolation model for AoA at constant $Ma = 0.845$ [-], RBF

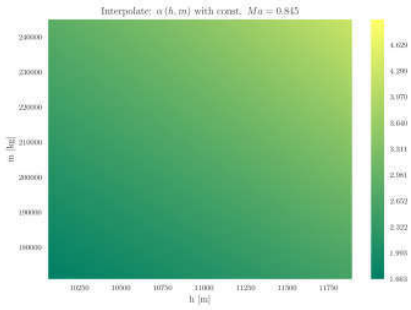

Figure 3.28: Surrogate interpolation model for AoA at constant $Ma = 0.845$ [-], Kriging

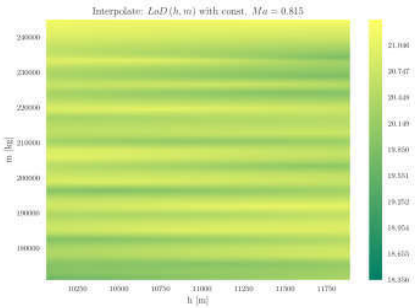

Figure 3.29: Surrogate interpolation model for LoD at constant $Ma = 0.815$ [-], RBF

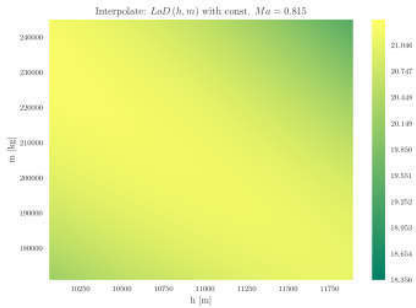

Figure 3.30: Surrogate interpolation model for LoD at constant $Ma = 0.815$ [-], Kriging

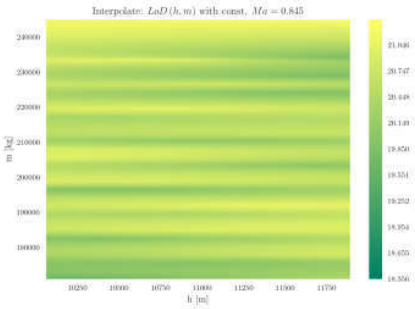

Figure 3.31: Surrogate interpolation model for LoD at constant $Ma = 0.845$ [-], RBF

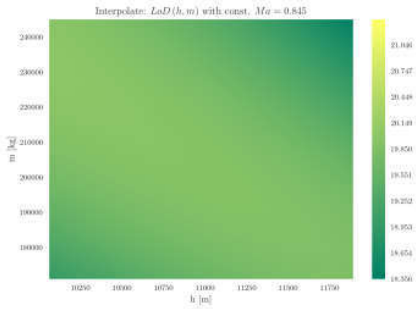

Figure 3.32: Surrogate interpolation model for LoD at constant $Ma = 11887.24$ [m], Kriging

The figures from 3.33 to 3.44 show the interpolations models, when a cut at constant mass is made. These are slices, where the RBF plots (left side) for the output variables $LoD, AoA, TSFC$ are also smooth. However, Kriging (right side) remains smooth, which suggests, that RBF and Kriging trends must not necessarily be contradicting. The second difference with regard to the constant Mach number and altitude slices is, that comparing RBF (left side) and Kriging (right side) small noticeable differences in the values for the output variables $LoD, AoA, TSFC$ in the overall space can be observed. This can be seen, e.g. by comparing figure 3.35 and 3.36. Consider the upper left corners and the whole upper edges. Here the clear contrast in the colors and thus in the values for AoA can bee seen. These differences, can also become extreme when comparing figures 3.43 and 3.44. Employing RBF the values for LoD are all greater or equal to $LoD \geq 21.046$. In contrast, the Kriging model (right side) also exhibits LoD values which are around 19. This highlights, the interpolation for LoD highly depends on the underlying surrogate interpolation model.

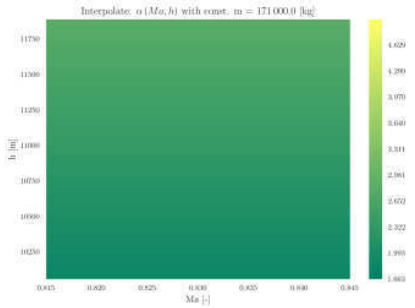

Figure 3.33: Surrogate interpolation model for AoA at constant mass $m = 171000$ [kg], RBF

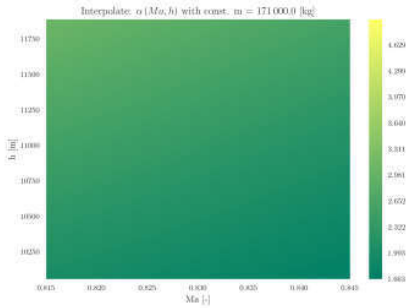

Figure 3.34: Surrogate interpolation model for AoA at constant mass $m = 171000$ [kg], Kriging

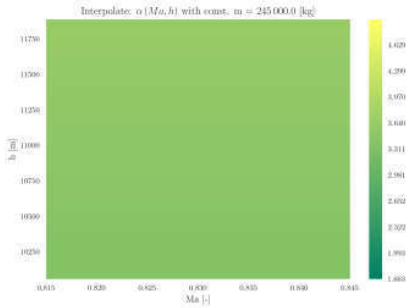

Figure 3.35: Surrogate interpolation model for AoA at constant mass $m = 245000$ [kg], RBF

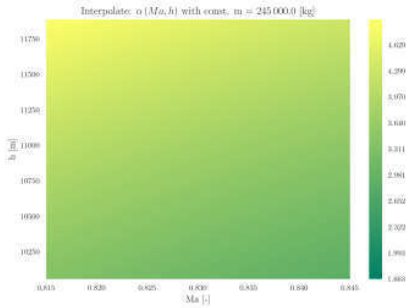

Figure 3.36: Surrogate interpolation model for AoA at constant mass $m = 245000$ [kg], Kriging

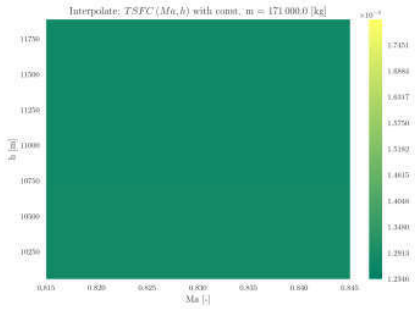

Figure 3.37: Surrogate interpolation model for $TSFC$ at constant mass $m = 171000$ [kg], RBF

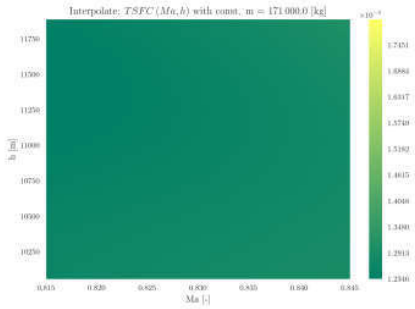

Figure 3.38: Surrogate interpolation model for $TSFC$ at constant mass $m = 171000$ [kg], Kriging

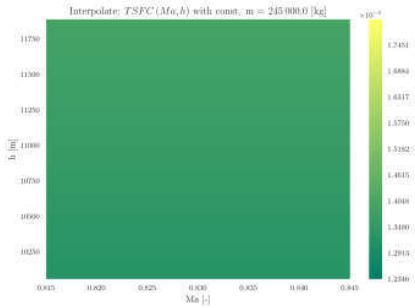

Figure 3.39: Surrogate interpolation model for $TSFC$ at constant mass $m = 245000$ [kg], RBF

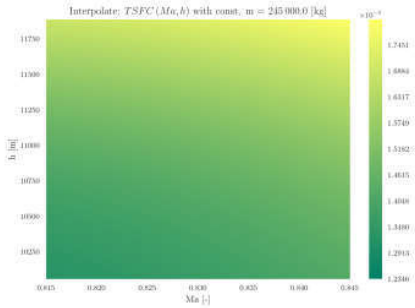

Figure 3.40: Surrogate interpolation model for $TSFC$ at constant mass $m = 245000$ [kg], Kriging

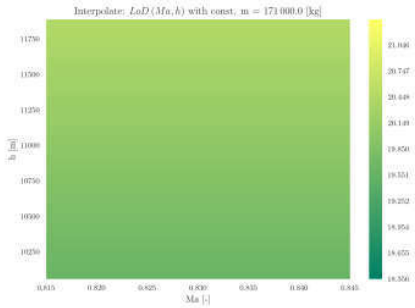

Figure 3.41: Surrogate interpolation model for LoD at constant mass $m = 171000$ [kg], RBF

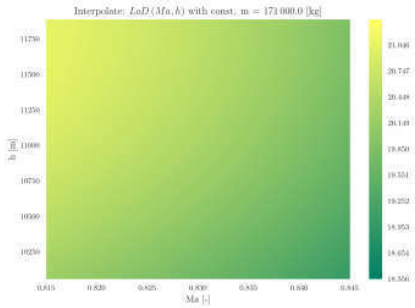

Figure 3.42: Surrogate interpolation model for LoD at constant mass $m = 171000$ [kg], Kriging

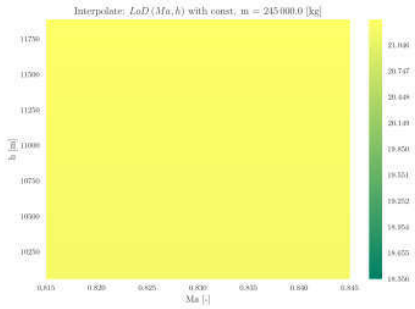

Figure 3.43: Surrogate interpolation model for LoD at constant mass $m = 245000$ [kg], RBF

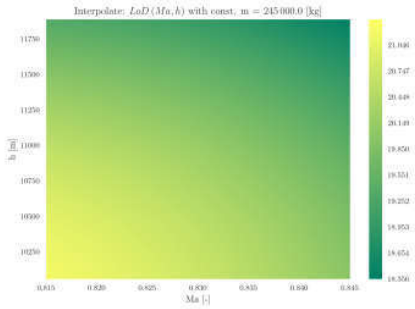

Figure 3.44: Surrogate interpolation model for LoD at constant mass $Ma = 11887.24$ [m], Kriging

This section can be concluded with the following. Clearly, there are differences in the outcome of the predicted values and trends for the desired output variables LoD, $TSFC$, AOA. As a consequence, once more it can be said that the choice of the surrogate model heavily determines the results of any further calculation.

3.4 Evaluating different surrogate models

In this section the different surrogate models Kriging (Gaussian), TPS and RBF are explored, which are all described in section 2.5. Since most explanations for the upcoming explorations were given in subsection 2.5.3, only necessary details will be repeated. The two missions which are going to be used for the following investigations are shown in table 2.3. Both missions passed the constraints check violation successfully and thus the number of the fix point iterations for calculating the gradients is low. The table 3.2 shows the options for Kriging and TPS and *SciPy*s RBF. For all *SciPy*s RBF kernels a *smooth* value of 0.1 [3] was used. More information about the settings and kernel is given in section 2.5.

***SMARTy* Kriging and TPS**			***SciPy***
Name	**Augmentation**	**Regularization**	**RBF Kernel**
A	-1	False	multiquadric
B	0	False	inverse
C	$+1$	False	gaussian
D	$+2$	False	linear
E	-1	True	cubic
F	0	True	thin plate
G	$+1$	True	-
H	$+2$	True	-

Table 3.2: Kriging, TPS and RBF parameter used for investigation surrogate quality

Similarly to the results sections before, much research has been done. However, not all results can be shown, rather results which are supposed to sum the found results up will be shown and elaborated. Recall, 35 effective or fully trimmed sample points were given as the input data for training the surrogate models. The investigations were performed for state and gradients as well. In subsection 2.5.3 three different versions were introduced, *V38, V28* and *V18*, where each version only effectively have three less sample points than its declarations name. According to subsection 2.5.3 for V38 the *Leave-one-out cross-validation* is applied. Whereas for V28 and V18 the *Root Mean Square Error* (RMSE) is calculated by having 10 and 20 additional sample points, respectively. For this additional sample points the true value of output variables AoA, LoD, $TSFC$ and their gradients is known.

The first exploration, which will be considered is by using the 38 sample data (effective 35 trimmed samples). For this V38, the total missions fuel mass m_f, the cruise fuel weight m_{cr} and the cruise starting mass m_s were monitored. However, since only for the cruise segment a physical based equation is applied, it is sufficient to only portray the curve of the total fuel mass m_f with the different surrogate models. As concluded in section 3.2, the indirect method is preferred to the direct method. Therefore, the indirect method is used in order to perform this surrogate investigation. Figure 3.45 and 3.46 exhibit the total fuel mass m_f

of mission 1, by employing Kriging and TPS, respectively. Figures 3.47 and 3.48 show the same for mission 2. The horizontal axis gives information about the chosen model intern parameter, which can be read from table 3.2. In both cases it can be observed, changing the model intern parameter does have an impact of the outcome (m_f). For both surrogate models with their different options the deviation is less than 260 kilogram. In case of Kriging and TPS in figures 3.45 and 3.48, respectively it is only 25 kilogram. Such a deviation is considered to be acceptable.

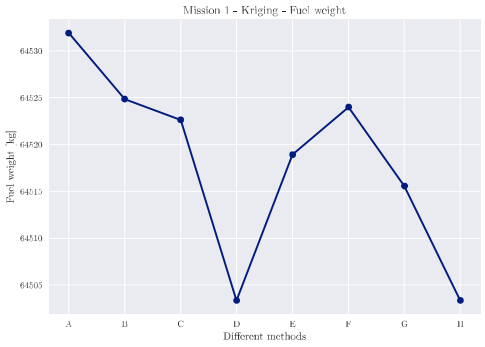

Figure 3.45: Final mission fuel weight m_f, Mission 1, Kriging variants

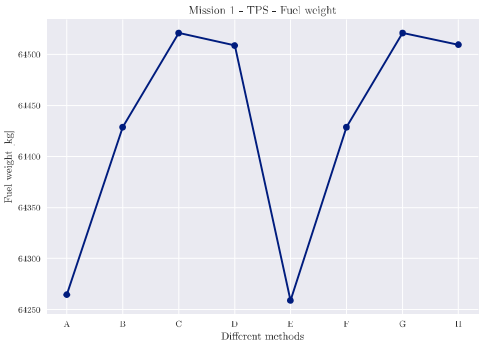

Figure 3.46: Final mission fuel weight m_f, Mission 1, TPS variants

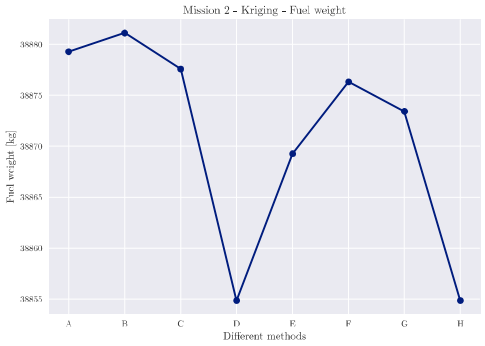

Figure 3.47: Final mission fuel weight m_f, Mission 2, Kriging variants

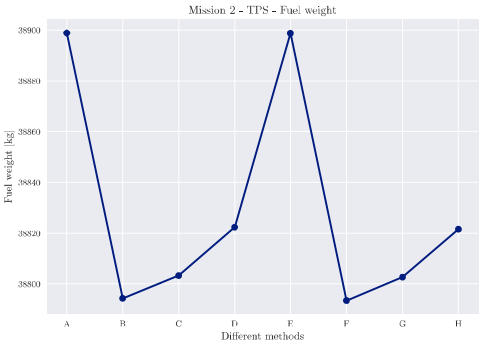

Figure 3.48: Final mission fuel weight m_f, Mission 2, TPS variants

Figures 3.49 to 3.52 exhibits the number of the fix point iterations, which were required for only mass calculation for the states, as explained in section 2.4. It can be seen that in no event more than 4 state fix point iterations are required. The difference between the gradient and the state fix point iterations can be simplified as follows. The gradient fix point iterations number is approximately twice as high as the state version. This is so, because the CDS evaluates the functions value at two points, thus the state fix point iteration is run twice.

Figures 3.53 to 3.56 depict the same kind of study for RBF and its options. It can be observed, that RBF predictions highly depend on the chosen kernel. Also, the number of the state fix point iterations are much higher. Figures 3.57 and 3.58 show a comparison with all surrogate models and their options for mission 1 and 2, respectively. It can be seen, that only

in case of mission 1, depending on RBF's kernel a match with Kriging and TPS occurs. In general, it can be stated that the *SciPy* RBF's output is noticeably different to *SMARTy*'s Krigings and TPS output. However, Krigings and TPS output exhibit a comparatively low deviation. Furthermore, it can be said that RBF has a much higher demand of state fix point iterations, which makes is unfeasible for application purposes.

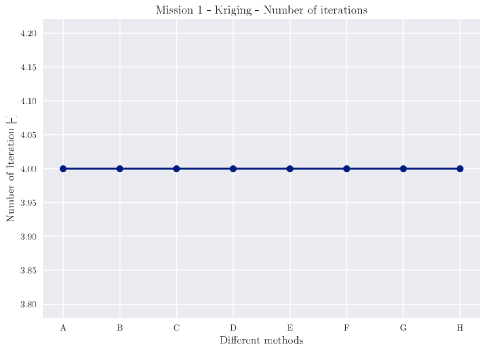

Figure 3.49: Number of the state fix-point iterations, Mission 1, Kriging variants

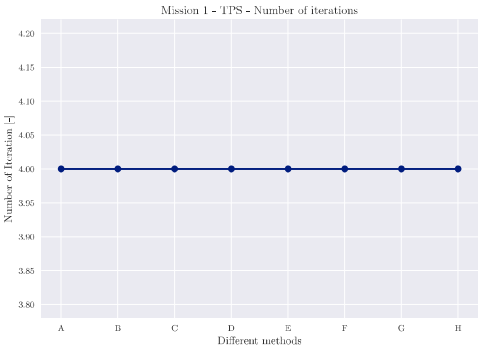

Figure 3.50: Number of the state fix-point, Mission 1, TPS variants

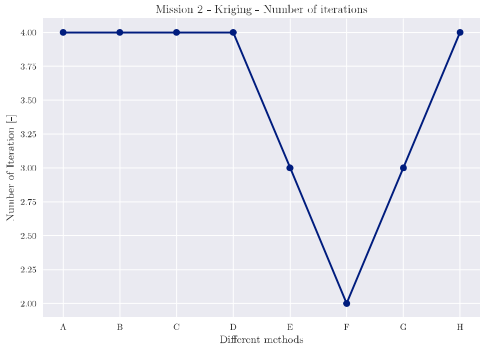

Figure 3.51: Number of the state fix-point iterations, Mission 2, Kriging variants

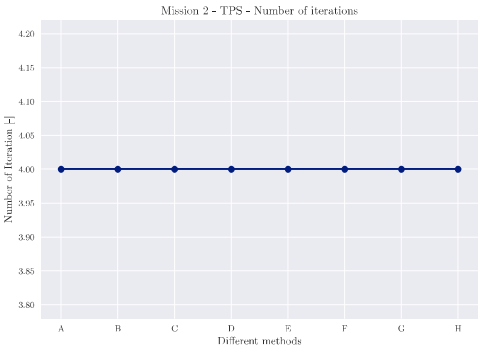

Figure 3.52: Number of the state fix-point, Mission 2, TPS variants

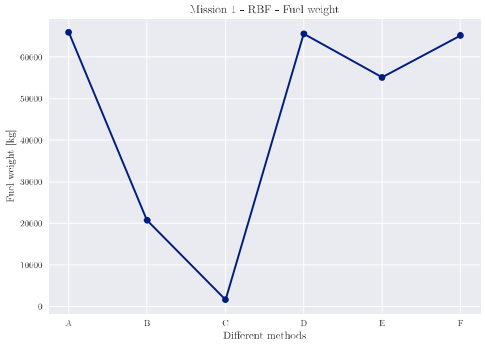

Figure 3.53: Final mission fuel weight m_f, Mission 1, RBF variants

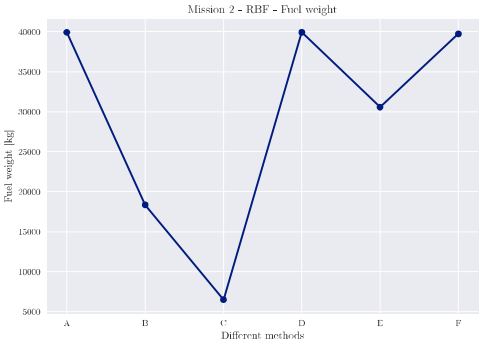

Figure 3.54: Final mission fuel weight m_f, Mission 2, RBF variants

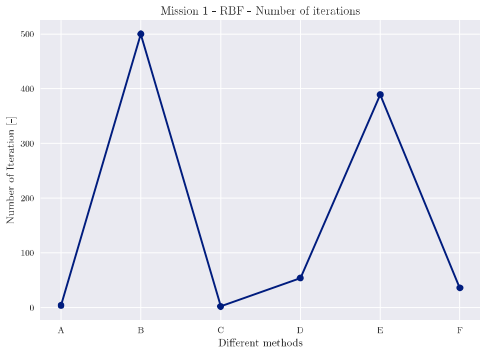

Figure 3.55: Number of state fix point iterations, Mission 1, Kriging variants

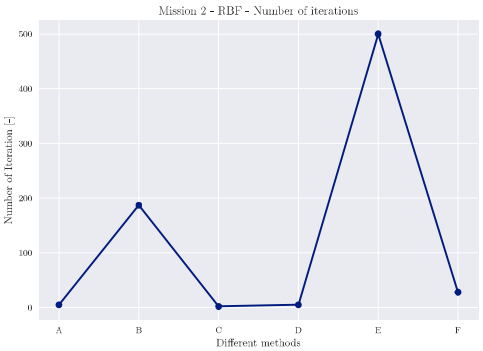

Figure 3.56: Number of state fix point iterations, Mission 2, TPS variants

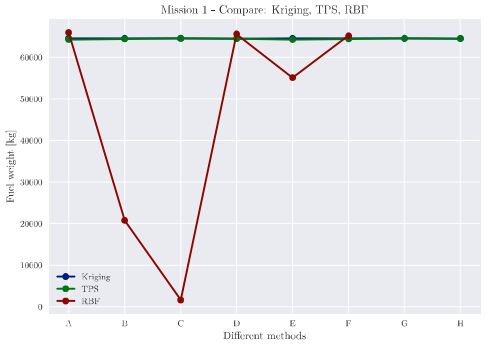

Figure 3.57: Total fuel mass m_f for all surrogate models, Mission 1

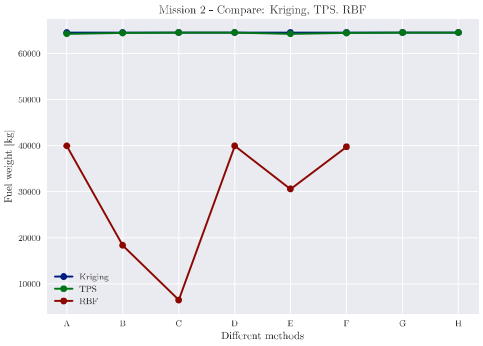

Figure 3.58: Total fuel mass m_f for all surrogate models, Mission 2

The next step is to consider the effect of a decreasing number of training samples and its effect on the surrogate models. This is done only for Kriging and TPS, since *SciPy*'s RBF showed a too high sensitivity on the chosen kernel and also required too many fix point

iterations. Kriging and TPS have the same 8 options. The total fuel mass m_f is plotted in the figures 3.59 to 3.62. Comparing m_f for mission 1 with Kriging and TPS via figures 3.59 and 3.60, respectively, the following can be observed. The vertical scaling axis of Kriging is finer, which means, a high change on the left figure results in a low change on the right figure (TPS). The maximum difference in the version which could occur for Kriging is 350 kg and for TPS is 1600 kg. Therefore, the absolute deviation needs to be calculated, which are shown in figures 3.61 and 3.62 for Kriging and TPS, respectively. In the mentioned figures, the green and red curves are more interesting to us, since it is assumed that V38 is the most accurate version and thus is considered as the basis. For mission 1, Kriging V18 and V28 are clearly closer to V38. In other words, Kriging tends to deliver with fewer data results closer to the fine sampling outputs than TPS. In case Kriging would also be the most accurate method (see subsection 3.4.1), it could be preferred for training surrogate models with a fewer number of training data.

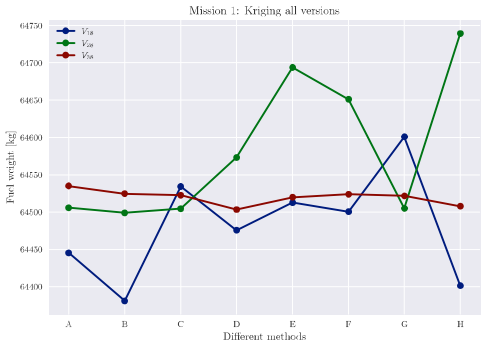

Figure 3.59: Total fuel mass m_f for all Versions, Mission 1, Kriging

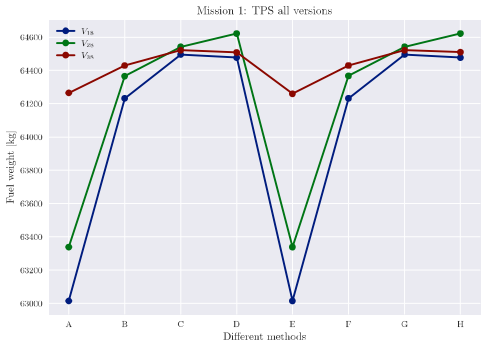

Figure 3.60: Total fuel mass m_f for all Versions, Mission 1, TPS

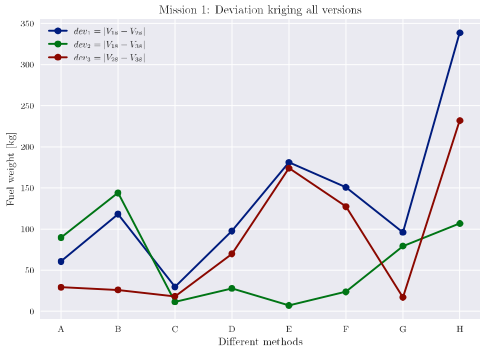

Figure 3.61: Deviations of the different versions of m_f, Mission 1, Kriging

Figure 3.62: Deviations of the different versions of m_f, Mission 1, TPS

The same also must be tested for mission 2. The results are presented in figures 3.63 to 3.66. The observations made for mission 1 are confirmed by having investigated on mission 2.

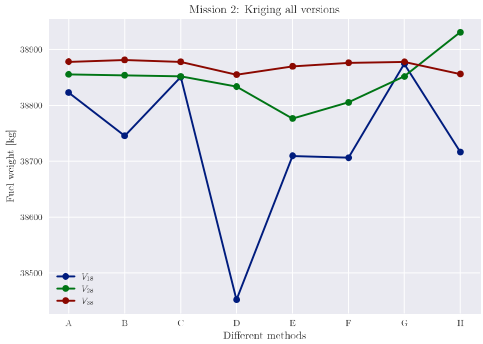

Figure 3.63: Total fuel mass m_f for all Versions, Mission 2, Kriging

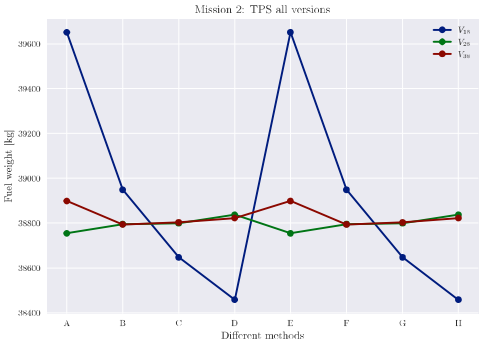

Figure 3.64: Total fuel mass m_f for all Versions, Mission 2, TPS

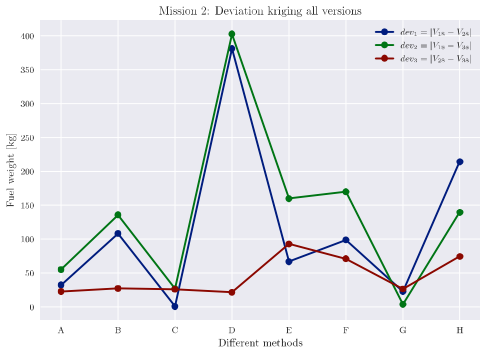

Figure 3.65: Deviations of the different versions of m_f, Mission 2, Kriging

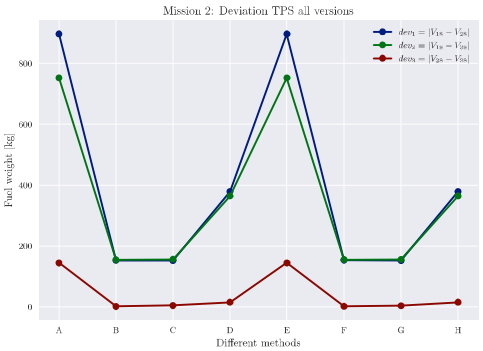

Figure 3.66: Deviations of the different versions of m_f, Mission 2, TPS

Up to now, only state solutions were discussed. However, the whole investigation process was also done for the gradients. Since the gradient investigation does not lead to further findings, only the most important results will be shown. Furthermore, only mission 1 is displayed as a representation of both missions The figures 3.67 and 3.68 depict the mean curve of the gradient of the total fuel masses w.r.t. all shape parameter. Mean curve means, that the results of the 8 options were used for calculating their respective mean values.

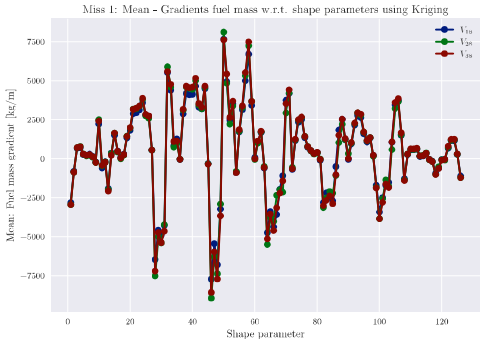

Figure 3.67: Total fuel mass gradients w.r.t. shape parameters, All versions, Mission 1, Kriging

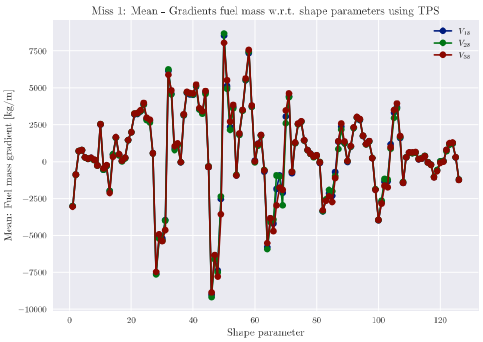

Figure 3.68: Total fuel mass gradients w.r.t. shape parameters, All versions, Mission 1, TPS

Next, the required execution time by having chosen the different surrogate models is going to be displayed. It is measured right after invoking *missioninformer* till the state and gradients for both missions were computed entirely and stored to the hard disk. In other words, it is the execution time of *missioninformer* itself, which is measured with different surrogate models. The results are depicted in figure 3.69. It can be observed that the Kriging versions are also more stable in the run time than TPS. The impact of different Kriging options does not have high impact of its execution time. TPS on the other hand, can be a little faster than Kriging with some options, however also can be much slower with other options. Also, depending on the sample data size, a high deviation can be observed, when regularization is activated (E to H).

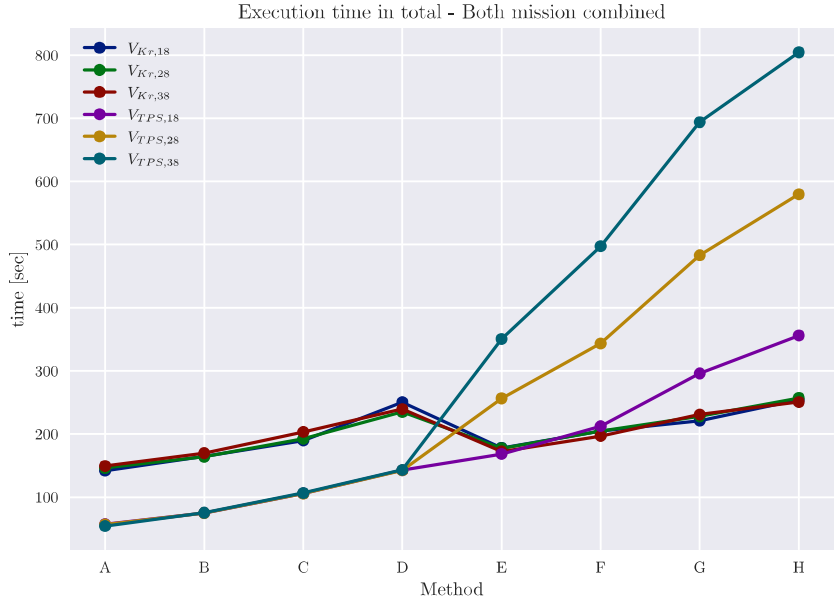

Figure 3.69: Missioninformers execution time with different surrogate models

3.4.1 Kriging and TPS accuracy

The objective of this subchapter is to present quantifiable information about the accuracy of Kriging and TPS models depending on their user-defined options. The question is, which model and which option leads to the lowest RMSE. The answer to this question is found using *SMARTy*'s automated model selection capability. The workflow for this purpose was repeated shortly in the introduction of this section and is given in more detail in subsection 2.5.3. Therefore, the results shall be presented straight forward. On the left side the best option set for the state is given. Here also the declaration of the 16 tested models is provided, which is valid for the gradient investigation, depicted on the right side, as well. For the state calculation, only three surrogate models $\tilde{LoD}, \tilde{AoA}, \tilde{TSFC}$ are generated. Therefore, out of the total 16 options for each of the three different surrogate models, only one offers the least RMSE. Therefore, for the states, each option only once can be found to be the best. In contrast, the gradient has multiple shape parameters and each shape parameter has his own best option. Due to this, one option can be found to have the least value of RMSE multiple times per associated output parameter $LoD, AoA, TSFC$.

The figure 3.70, 3.72 and 3.74 depict the state version for V18, V28 and V38 respectively. The figures 3.71, 3.73 and 3.75 their respective gradient version. Each figure on the left side contains a legend, which is valid for both sides. A clear distriction between Kriging and TPS models can be made with the letters *A* - *H* and *I-P*, respectively. Unfortunately, no absolutely clear winner can be filtered out of these results. The distributions are too different for this purpose. However, a suggestion can still be made by starting with options, which should not be taken into consideration. Viewing all the gradient results (right side), options *C, I, J, K* and *L* show the least number of best RMSE values. Also, the same options were not found to be the best choice for the states. The option *A* is also not to be found as the best choice for states nor does it have a good result for the gradients for V18 and V28. Option *A* is clearly no suggestion, but does not perform as bad as the other options mentioned before. The remaining possible options are: *B, D, E, F, G, H, L, M, N, O* and *P*. From these options *F, H, M* and *N* exhibit the best state results. These are 2 Kriging and 2 TPS models. 3 out of these 4 models include regularization and have the augmentation value of 1. From here on, the most appealing option by including the gradients, is option *N*. Furthermore, figures 3.62 and 3.66 should be considered in this examination. They depict the deviation for the different versions (V18, V28, V38) for *SMARTy*'s TPS with an augmentation of -1. For both missions the deviation is high. This option is declared as *J* in the figures 3.70 and 3.75 and also here, unsatisfactory results are found.

In conclusion, it can be said, no real winner could be found out for the options. However, using regularization and an augmentation of value 1 can be recommended. If a final decision had to be made, option *N*, the TPS model with regularization and an augmentation of 1 should be chosen. Note that performing this model selection study took more than 3 days on the workstation described in section 1.1. This means performing model selection for each new set of training samples within an aerodynamic optimization is not feasible. Therefore, TPS with active regularization and an augmentation of 1 is set as the default model in *missioninformer*.

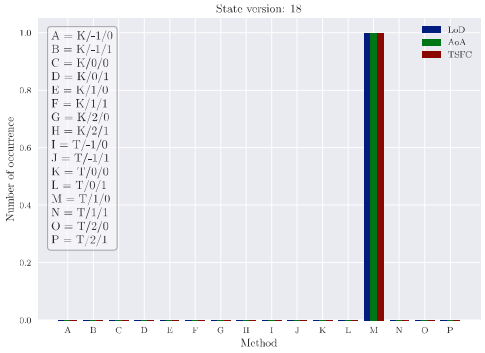

Figure 3.70: Lowest RMSE by having compared Krigings and TPS 8 options, state V18

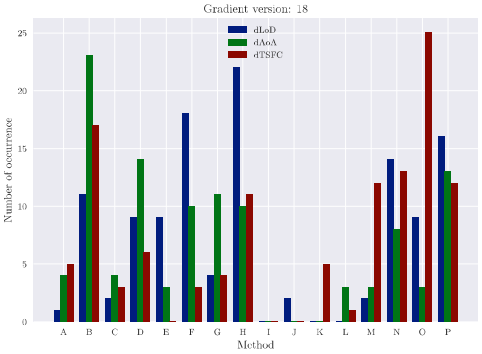

Figure 3.71: Lowest RMSE by having compared Krigings and TPS 8 options, gradient V18

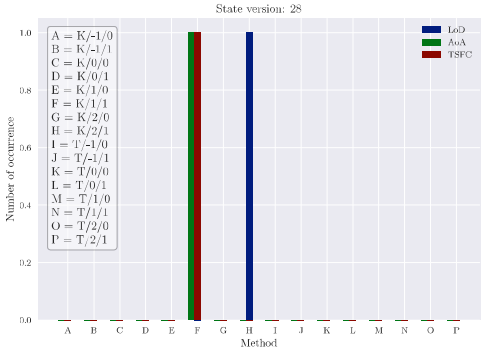

Figure 3.72: Lowest RMSE by having compared Krigings and TPS 8 options, state V28

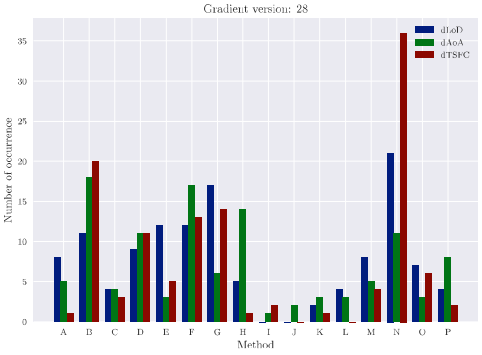

Figure 3.73: Lowest RMSE by having compared Krigings and TPS 8 options, gradient V28

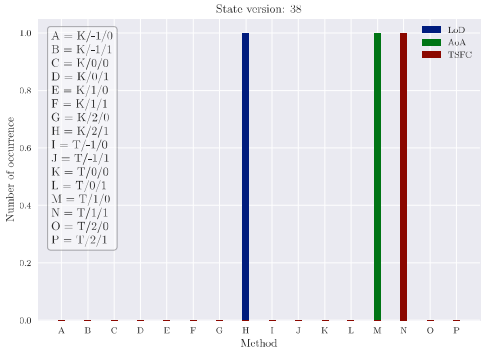

Figure 3.74: Lowest RMSE by having compared Krigings and TPS 8 options, state V38

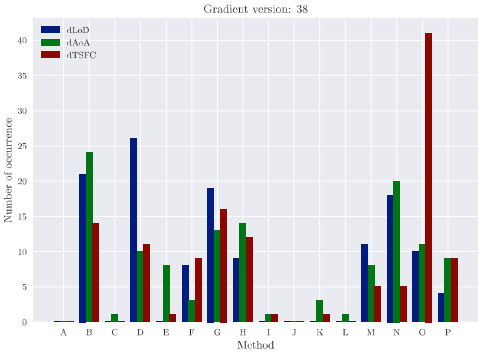

Figure 3.75: Lowest RMSE by having compared Krigings and TPS 8 options, gradient V38

4 Discussion

The *missioninformer*, a tool, was proposed to help to fulfill the environmental demands for reducing air traffic pollution. Clearly, the aircraft sector contributed and contributes to the current environmental crisis which we are facing. Since an aircraft has become indispensable, not only for the individual, but also for the global economy, the solution for a healthier environment cannot be banning aircraft, but rather improving it with regard to CO2, NOx and SO2 emissions. The *missioninformer*, a lightweight tool was written from scratch. Its major task is to calculate the consumed fuel for one or multiple individual missions for one aircraft. For the most important part of the mission, the cruise segment, physics-based equations are applied. In section 2.3 they are introduced, the precision with which they are solved is satisfactory and is presented in section 3.1. They are fed with interpolated aerodynamic data, where the interpolation models were generated by using high fidelity RANS-based training samples.

The next step *missioninformer* went, was also to include the remaining flight segments by employing fuel fractions. All the used fuel fractions were given in section 2.2 in table 2.1. At this stage, the *missioninformer* can predict the fuel required for a whole flight mission. For real world applications, the user must be able to define the flown mission himself. The *missioninformer* meets that need and in a structured way, so less effort is required, which is described in section 1.3

Since the *missioninformer* is a leightweight code and because it can be used as a black box, it already can be implemented easily into a global gradient free optimization workflow. Since gradient-based optimization is known to be faster, especially with a high number of design variables, *missioninformer* was enhanced with the ability to calculate gradients. These are gradients of the fuel mass of the whole flight mission with respect to all arbitrarily given shape parameters. To guarantee a high accuracy of *missioninformer*'s state and gradient solutions many different studies were conducted.

In order to solve the ODE for the cruise flight segment, two tested solvers exhibited particular high accuracies in their results. One of them ($RK45$) also demanded few integration steps and thus excels in terms of execution time. Therefore, ($RK45$) is well suited for research and industrial applications. Based on this reasoning it is used as a standard ODE solver.

For the gradient calculation, by employing central differencing, it can be stated that the iterative indirect method was found to be more stable. Also, choosing the step-size of $1e{-}5$ could be proven to be optimal. This value neither leads to numerical instabilities nor does it not comply with the accuracy requirements. Therefore, the default method for solving the gradients with respect to the shape parameters is the indirect method with its default step-size of $1e{-}5$.

Having compared *SciPy*'s RBF and *SMARTy*'s Kriging (Gaussian) and TPS interpolation

models for different model options, the following can be stated. Clearly, there are differences in the outcome of the predicted values and trends for the desired output variables $LoD, TSFC, AOA$. As a consequence, once more it can be said that the choice of the surrogate model heavily determines the results of any further calculation.

After having performed a computational intensive model selection investigation, the following conclusion is made. No real winner could be found out for the options. However, using regularization and an augmentation of value 1 can be recommended. If a final decision had to be made, the TPS model with regularization and an augmentation of 1 should be chosen. Note that performing this model selection study took more than 3 days on the workstation described in section 1.1. This means performing model selection for each new set of training samples within an aerodynamic optimization is not feasible. Therefore, TPS with active regularization and an augmentation of 1 is set as the default model in *missioninformer*.

For the sake of applicational feasibility, execution time for the *missioninformer* with different surrogate models and their respective different intern parameters was measured. With that it can be stated, the *missioninformer* suits well for a fast and easy implementation into a gradient-based and gradient-free optimization workflow, where it completes its calculation within few minutes. Having introduced the *missioninformer* broadly, it clearly can be considered as a possible tool at hand to be integrated into new digital design methods to reduce the negative impact of aviation on the environment, which can not be neglected nor overlooked.

5 Conclusions and outlook

A tool for calculating the total fuel mass and its gradient w.r.t. arbitrary shape parameter and for any user defined mission or multiple missions is presented. It was generated from scratch and can be used as black box. It could be observed, that in case of two missions, the runtime was between 50 to 800 seconds (figure 3.69) depending on the employed interpolation models and including the gradient computation. The following surrogate models were intensively investigated with different options: RBF from the *SciPy* library, Kriging (Gaussian) and TPS from *DLR*'s toolbox *SMARTy*. RBF could clearly be filtered out from the list of the attractive surrogate model options, mainly due to its long execution times and highly options dependent results. In order to solve the ODE, which is required for computing the fuel fraction of the cruise segment, different numerical solvers were used. By having applied simplifications an analytical solution of the ODE was introduced. The analytical solution was compared with different numerical solvers solutions. With these comparisons, statements about the accuracy of the numerical ODE solver could be made. The solution of the total fuel mass incorporates mass snowball effects though the explained fix-point iteration. For the gradients, also an analytical approach was desired. However, after comparing the analytically determined gradients with central finite differences results, the assumption was made for the derivation were found to be not tenable. A detailed analysis for finding an appropriate value for the step-size of the central differencing scheme was undertaken. Its results could be shown to be unambiguous.

As an outlook, the analytical solution for the gradients should be revisited. It failed due to the elaboration of the mass term. It is assumed that there is a way to reformulate this term and thus obtain the gradients analytically. An alternative could be employing automatic differentiation. Up to now, only the most important flight segment, the cruise segment is described by equations which are based on physical observations. The currently used fuel fractions for the other flight segments could be replaced by physics-based equations as well. Also in practice, step-climbs are performed in order to fly with an optimal LoD. This means, the assumption of constant altitude, which is made in *missioninformer* needs to be adapted. Next, state and gradient calculations are not consistent due to the chosen approach with multiple independent surrogate models. *Missioninformer* should be tested in an aerodynamic gradient-based optimization to find out how many training points are necessary to achieve significant reduction in the missions fuel consumption. Finally, the gradients of the missions fuel w.r.t. the structural mass needs to be calculated in order to use *missioninformer* within a multi-disciplinary-optimization workflow.

Bibliography

[1] Homepage - SUAVE, 2021.

[2] NumPy's allclose, August 2021.

[3] RBF - SciPy, August 2021.

[4] SciPy's ODE solver, August 2021.

[5] M. Ahmed and N. Qin. Surrogate-Based Aerodynamic Design Optimization: Use of Surrogates in Aerodynamic Design Optimization. *International Conference on Aerospace Sciences and Aviation Technology*, 13(AEROSPACE SCIENCES):1–26, May 2009.

[6] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[7] Benoit Dabas, Nathalie Bartoli, Thierry Lefebvre, Francois Gallard, Anne Gazaix, Thierry Y. Druot, and Damien Guénot. Error-Based Adaptive Coupling Process Between Multipoint High-Fidelity Aerodynamics and Mission Performance for Shape Optimization in the MDA-MDO Project. In *AIAA Aviation 2019 Forum*, Dallas, Texas, June 2019. American Institute of Aeronautics and Astronautics.

[8] Michael Eldred, Anthony Giunta, and S. Collis. Second-Order Corrections for Surrogate-Based Optimization with Model Hierarchies. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, August 2004. American Institute of Aeronautics and Astronautics.

[9] Richard C Feagin and William D Morrison. Delta method, an empirical drag buildup technique. 1978.

[10] Stefan Goertz, Mohammad Abu-Zurayk, Caslav Ilic, Tobias F. Wunderlich, Stefan Keye, Matthias Schulze, Christoph Kaiser, Thomas Klimmek, Özge Süelözgen, Thiemo Kier, Andreas Schuster, Sascha Daehne, Michael Petsch, Dieter Kohlgrüber, Jannik Häßy, Robert Mischke, Alexander Weinert, Philipp Knechtges, Sebastian Gottfried, Johannes Hartmann, and Benjamin Fröhler. Overview of Collaborative Multi-Fidelity Multidisciplinary Design Optimization Activities in the DLR Project VicToria. In *AIAA AVIATION 2020 FORUM*, VIRTUAL EVENT, June 2020. American Institute of Aeronautics and Astronautics.

[11] Justin S. Gray, John T. Hwang, Joaquim R. R. A. Martins, Kenneth T. Moore, and Bret A. Naylor. OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, April 2019.

[12] EM Greitzer. Design Methodologies for Aerodynamics, Structures, Weight, and Thermodynamic Cycles. *Cooperative Agreement No. NNX08AW63A*, 2010.

[13] S. Görtz, \textbackslashvC Ilić, M. Abu-Zurayk, R. Liepelt, J. Jepsen, T. Führer, R. Becker, J. Scherer, T. Kier, and M. Siggel. Collaborative Multi-Level MDO Process Development and Application to Long-Range Transport Aircraft. 2016.

[14] Caslav Ilic. Goal function in Digital-X. Braunschweig, Germany, September 2013. DLR.

[15] Jason Y. Kao, John T. Hwang, Joaquim R. R. A. Martins, Justin S. Gray, and Kenneth T. Moore. A Modular Adjoint Approach to Aircraft Mission Analysis and Optimization. In *Proceedings of the AIAA Science and Technology Forum and Exposition (SciTech)*, Kissimmee, FL, January 2015.

[16] Daniel G Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951. Publisher: Southern African Institute of Mining and Metallurgy.

[17] D.S. Lee, D.W. Fahey, A. Skowron, M.R. Allen, U. Burkhardt, Q. Chen, S.J. Doherty, S. Freeman, P.M. Forster, J. Fuglestvedt, A. Gettelman, R.R. De León, L.L. Lim, M.T. Lund, R.J. Millar, B. Owen, J.E. Penner, G. Pitari, M.J. Prather, R. Sausen, and L.J. Wilcox. The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018. *Atmospheric Environment*, 244:117834, January 2021.

[18] Rhea P. Liem, Gaetan K. W. Kenway, and Joaquim R. R. A. Martins. Multimission Aircraft Fuel-Burn Minimization via Multipoint Aerostructural Optimization. *AIAA Journal*, 53(1):104–122, January 2015.

[19] Rhea P. Liem, Charles A. Mader, Edmund Lee, and Joaquim R. R. A. Martins. Aerostructural design optimization of a 100-passenger regional jet with surrogate-based mission analysis. In *2013 Aviation Technology, Integration, and Operations Conference*, Los Angeles, CA, August 2013. American Institute of Aeronautics and Astronautics.

[20] Rhea P. Liem, Charles A. Mader, and Joaquim R.R.A. Martins. Surrogate models and mixtures of experts in aerodynamic performance prediction for aircraft mission analysis. *Aerospace Science and Technology*, 43:126–151, June 2015.

[21] Joaquim R. R. A. Martins and John T. Hwang. Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models. *AIAA Journal*, 51(11):2582–2599, November 2013.

[22] Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963. Publisher: Society of Economic Geologists.

[23] LA McCullers. Aircraft configuration optimization including optimized flight profiles. 1984.

[24] Martin Meckesheimer, Andrew J. Booker, Russell R. Barton, and Timothy W. Simpson. Computationally Inexpensive Metamodel Assessment Strategies. *AIAA Journal*, 40(10):2053–2060, October 2002.

[25] Andrei Merle, Caslav Ilic, Mohammad Abu-Zurayk, Jannik Häßy, Richard-Gregor Becker, Matthias Schulze, and Thomas Klimmek. High-Fidelity Adjoint-based Aircraft Shape Optimization with Aeroelastic Trimming and Engine Coupling. September 2019.

[26] Ruben Perez and Joaquim RRA Martins. pyACDT: An object-oriented framework for aircraft design modelling and multidisciplinary optimization. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 5955, 2008.

[27] Jan Roskam. *Airplane design*. DARcorporation, Lawrence, Kan, 1986.

[28] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4), November 1989.

[29] Timothy W Simpson, Timothy M Mauery, John J Korte, and Farrokh Mistree. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA journal*, 39(12):2233–2241, 2001.

[30] Egbert Torenbeek. *Synthesis of Subsonic Airplane Design*. Springer Netherlands, Dordrecht, 1982.

[31] Felipe A. C. Viana, Timothy W. Simpson, Vladimir Balabanov, and Vasilli Toropov. Special Section on Multidisciplinary Design Optimization: Metamodeling in Multidisciplinary Design Optimization: How Far Have We Really Come? *AIAA Journal*, 52(4):670–690, April 2014.

DLR-IB-AS-BS-2021-136

Development of a module for mission analysis for a gradient-based aerodynamic shape optimization process

Javed Butt

Verteiler:

Institutsbibliothek	1 Exemplar
Verfasser	3 Exemplare
Institutsleitung	1 Exemplar
Abteilungsleiter	1 Exemplar
Deutsche Bibliothek in Frankfurt/Main	2 Exemplare
Niedersächsische Landesbibliothek Hannover	1 Exemplar
Techn. Informationsbibliothek Hannover	1 Exemplar
Zentralbibliothek BS	2 Exemplare
Zentralarchiv GÖ	1 Exemplar
Reserve	5 Exemplare